UPC - FIB
Facultat d'Informatica de Barcelona

Informe

Final career project at the CNES (National Space Study Centre) -
Toulouse Spatial Centre

Study of the Lustre file system performances before its installation in a computing Cluster

Yann Chanel
12 rue des Filatiers
31000 TOULOUSE
France
Tél. : (+33)5 61 32 85 14 / (+33)6 08 50 89 32
yann.chanel@ensimag.imag.fr

Option « Telematic Networks and Operating Systems »

18/02/2008 – 18/08/2008

CNES
Centre Spatial de Toulouse
18 rue Edouard Belin
31401 Toulouse Cedex 9
FRANCE

Internship responsible
M. Bertrand Legras
Academic tutor
M. Josep Lluís Larriba Pey

Contacts

Anyone trying to contact the student can refer to the coordinates on the first page of this report.

The internship manager can be contacted at the following address, telephone number, fax or e-mail :

Bertrand LEGRAS - BPi 3517
Centre National d'Etudes Spatiales
18 rue Edouard Belin
31401 TOULOUSE Cedex 9
France

Tel: (+33)5 61 27 32 81 / Fax: (+33)5 61 28 18 93
E-mail: bertrand.legras@cnes.fr

Table of contents

1. Abstract

The subject of this internship report is the performance evaluation of a Cluster file system called Lustre, before its installation into a computing Cluster, or other computing systems. Performance evaluation includes benchmarks, comparison to other file systems (ext3, NFS), and study of tuning possibilities.

In this report we first present the file system Lustre, and then we explain how the tests were driven, including the methodology of the tests, the installation of the test platform, and a preliminary research of the main bottlenecks.

Three types of benchmarks have been driven: read/write of big files, including both sequential and random access, in order to test the throughput; read/write of small files, in order to test the cache efficiency; and creation/deletion of empty files, in order to test the access times.

The aim is to see in which cases it is convenient to use the Lustre file system.

Keywords: Lustre, ext3, NFS, file system, benchmark, access time, throughput, Cluster, Storage Area Network (SAN).

2. Acknowledgements

I worked in a very pleasant team, in which I was well integrated and with whom I had a lot of fun. For this I would like to thank the chief of the service François Jocteur-Monrozier, and all my workmates: Corinne Michau, Stephanie Marel, Bruno “Rabbit” Vidal, John Moyard, Pascal Richard, Michel Hummel, and, last but not least, Philippe Cam-Halot.

I would like to thank also Cathy Mazier, who made my life at the CNES easier by helping me for a lot of peanuts, my school tutor Franck Rousseau, who gave me some advice about this report, Celia Picard and Nicole Levy, who spent some time reading and correcting this report.

A very special thank to my CNES tutor Bertrand Legras who took care of me, and tried to do his best for me, despite of all the work he had.

3. Introduction

3.1. Aim of the internship

The aim of the project is to study a new file system that will be used in a computing Cluster, and to compare it to others already in use at the CNES. Hence, the project comes in the direct line of the need to be aware of new technologies.

The file system to study is a Cluster file system called Lustre, and its documentation is available at http://www.clusterfs.com. The project should start by the installation of the file system on a test platform. Once this step is done, a performance testing protocol must be defined and then executed on this system. These tests must include representative benchmarks.

As these tests may not be really representative of the real systems use, some tests must be carried out on the system in use. Then the same tests should be run on others file systems like Network File System[1] (NFS) and General Parallel File System (GPFS), which are both already in use at the CNES, and results must be compared to those obtained from Lustre.

Moreover, it will be interesting to compare the Lustre file system to ext2 and ext3, which are the most common file systems, and then compare the Lustre protocol to other common protocols allowing file sharing on remote file servers: NFS and FTP.

The final report shall include the tests results and their interpretation, and provide recommendations to choose the right solution and the best configuration possible.

3.2. Summary

Now we have explained the aim of the internship, we will introduce the CNES in paragraph 4.

Our work will be first explained first in paragraph 5, with the applied methodology and the objectives. Paragraph 6 explains the Lustre mechanism, and its installation on the test platform. Then, paragraph 7 explains the configuration of the test platform, and paragraph 8 reports a preliminary study of the test platform, and the modification to bring in order to eliminate the main bottlenecks.

Then come the different results. We will present the tests results on both test platform and computing systems, on paragraphs 9 and 910. Paragraph 11 is about the file system tuning, explaining what can be tuned, and how to gain efficiency. A synthesis of these tests is given in paragraph 12.

At last the time repartition of the internship will be explained in paragraph 13, and the last paragraph will be dedicated to a conclusion about the internship.

4. The CNES

4.1. Overview

The National Centre for Spatial Studies (CNES) was founded in 1961. It is a public agency, in charge of shaping, and implementing the French spatial policy inside the European Union.

Its role also consists in the coordination of the French participation to the European Spatial Agency (ESA), for which it designed and tested the Ariane rockets.

Its budget is around 1 700 M€, of which almost 40% goes to the ESA.

The CNES employs around 2500 workers, dispensed over 4 different sites: Kourou (in French Guyana), Paris and Evry, each one with around 250 employees, and Toulouse, the biggest one with more than 1700 employees.

[1] Documentation available at http://nfs.sourceforge.net

4.2. The Toulouse Space Centre (CST)

There are around 2500 people working at the CST, including CNES employees, affiliates companies employees and external firms employees (like Thalès, IBM, Sun Microsystems, etc).

The activity mainly consists in the management and the exploitation of the current projects and the preparation of the future ones.

The activity of the CST is divided in six missions:

1. Prepare the programs of the future
2. Bring support to the development of space techniques
3. Lead the programs being developed
4. Exploit the operational systems
5. Carry out some heavy means
6. Teaching and knowledge spreading

Part of the last one is accomplished through internships.

4.3. The DSI

The role of the Direction of Computer Science System (DSI) is to provide a computer science support to the internal users of the information system, and to offer them a big variety of technical services.

The Exploitation and Architecture sub-direction is part of the DSI, and it is composed of five departments, one of them being the Computer Science Architecture service. That is the service I worked in, with 16 other employees.

The mission of the people working in this service is to define the ground systems architectures and to implement them. For each project, their role is to find some architectural solutions, then to study their efficiency, their cost, and technical specifications, and once an architecture is chosen, to organize its exploitation with other services of the CNES.

To offer the best solutions, the computer science architect must be aware of new technologies.

5. Methodology and tests requirements

5.1. Methodology

At the beginning, we will have to install our test platform – hardware and operating system. Then we will try to understand how does Lustre work, and install it onto the platform. Once all this will be done, we will be able to start testing.

In a first time, we will try to detect the main bottlenecks of the system, and see how we can upgrade it to get better performances. On a second time, we will test the cache, throughput and access times of each file system, to find out which one is better for small files (low access times), and which one is better for large files (high throughput).

Then we will perform the same tests on the computing systems of the CNES. These tests will have two objectives: the first one is to compare the performances on the test platform to the

ones of the computing systems, to find out if the test platform results are valid for the “real life”; the second objective is to test the different file systems that are installed on the computing systems, to determine advantages and disadvantages of each one.

At last, we will perform tests on Lustre, changing its configuration, to try to determine the best value for some parameters.

5.2. Comparison tests

The objective of these tests is to determine the performances of some file systems, and to compare them to Lustre performances. To achieve this, it will be necessary to execute exactly the same test series on every file system. The file systems to be compared with Lustre are ext3, for local tests, and NFS, for remote tests. Both are already in use on CNES computing systems.

Three kinds of tests will be driven:

- Access time tests, consisting of creating empty files, and removing them
- Throughput tests, which must be performed for the basic operations permitted on any file system (read, write, etc.), and must also cover the utilizations made by the clients of the file servers, that is, perform sequential and random access. Files for these tests must be big enough to avoid any cache effect. Since the clients and servers RAM is 4 GB, we will perform the tests on 8 GB files.
- Cache efficiency tests, consisting basically in the same tests as throughput tests, with smaller files, so they can fit in the cache.

Moreover, the tests must be driven from 3 different user points of view:

- Local: the tests will be performed on the file server itself.
- Single client: the operations will be performed through the network, on a client that will be the only one accessing to and working on the file server.
- Concurrent: several clients performing the tests simultaneously, so that they will access the files concurrently.

The tests will be run with “basic” file systems configurations. That means that none of the file systems will be tuned after installation. Thus, it will be easier to perform again the same tests later, for instance on another hardware, and to compare the results to those exposed in this report.

The tools used for these tests will be Iozone, a file system benchmark tool, and the Unix ‘**`dd`**’ command for throughput and cache efficiency tests ; and the Unix ‘**`touch`**’ and ‘**`rm`**’ commands for access time measurements. All these tools are presented in the Appendix I.

5.3. Cluster tests

These tests consist in evaluating the performances of servers in production at the CNES. To achieve this, an account on different computing systems will be required.

The objective of such tests is to compare the results obtained in the previous phase to the one that can be obtained on the computing systems – reminding that these systems are used by many people at the same time, including during the tests.

The test methodology for these tests will be exactly the same as for the test platform, to make the comparisons easier.

5.4. Configuration tests

These tests must be run with different Lustre software and hardware configurations. The objective is to find the best configuration, depending on the use that will be done of the file system.

The idea is to compare, for each configuration, the test results to the ones obtained from the "basic" configuration.

The parameters to test are basically those described in the chapter named "Lustre tuning" of the Lustre Operations Manual[9]. Another parameter that must be included in the tests is the I/O scheduler, as suggested in the Lustre Operations Manual[10].

Additionally, we shall test different configurations of Lustre, trying to put all the services on the same machine, or trying to separate them. These services are explained in paragraph 6.3. At last, we can try to check the impact of using one or several data servers.

6. The Lustre File System

6.1. Overview and objectives

Lustre is a distributed file system designed for large scale Cluster computing, available under the GNU General Public License[2]. It was created by the Cluster File System company, and designed for Linux systems. It is now maintained by Sun Microsystems[3], since this one acquired Cluster File System, on October 2, 2007, with the objective of adding support for Solaris systems [4].

According to Sun Microsystems, the objective of Lustre is to provide a scalable file system to meet the increasing demands of high-performance clusters. Thus, Lustre is currently designed to support "tens of thousands of nodes, petabytes of data, and billions of files", and this capacities are going to be expanded in a near future[5].

6.2. Lustre version

All the tests have been driven on the version 1.6.4.2 of Lustre, which was the latest available version at the beginning of the internship, mid-February 2008. New versions (1.6.4.3, 1.6.5) have been delivered later in 2008. Version 1.6.4.3 contains just some corrections, whereas version 1.6.5 contains some interesting enhancements, among which the most important is a great improvement of NFS writes speed on 2.6 kernels.

To keep results homogeneity, we kept running with the version 1.6.4.2 for all the internship. However, we strongly recommend performing later the same tests as those driven in this report on the latest Lustre versions, to notice the improvements. Be careful, on the version 1.6.5, an NFS-like root squash functionality has been added, that limits the rights for root accesses. As we performed all the tests with root rights, some tests may not work on versions newer than 1.6.5.

All new versions of Lustre File System can be downloaded from the Sun Microsystems website[6], and the Change Log for each version can be found on the Lustre wiki website [17].

6.3. Lustre Clusters

Lustre file systems consist of 3 major components:

- The Management Server (MGS) defines the configuration of all the Lustre file systems in a company. It gathers information from the others servers of the file systems, and provide it to the clients.
- The Meta Data Targets (MDT) provide meta data storage for a single Lustre file system. Meta data are basically hierarchy, files attributes, permissions, etc.
- The Object Storage Targets (OST) provide storage for the data.

A single MGS can manage several Lustre file systems. Its disk can be shared only with a single MDT.

Several MDTs can run on a single Meta Data Server (MDS). In the same way, a single Object Storage Server (OSS) can host several OSTs. Both MDS and OSS servers can work in pair through a failover mechanism. Thus, the load is balanced between the 2 servers, and when one of the servers breaks down the other can assume the entire work.

Nowadays, only 2 MDS can be running simultaneously in a Lustre file system, but the improvements of the next versions of Lustre will permit up to 256 MDS at the same time. On the contrary, a Lustre file system can host up to 1000 OSS, and 10,000s of clients.

Figure 1 gives an example of complex Lustre configuration.

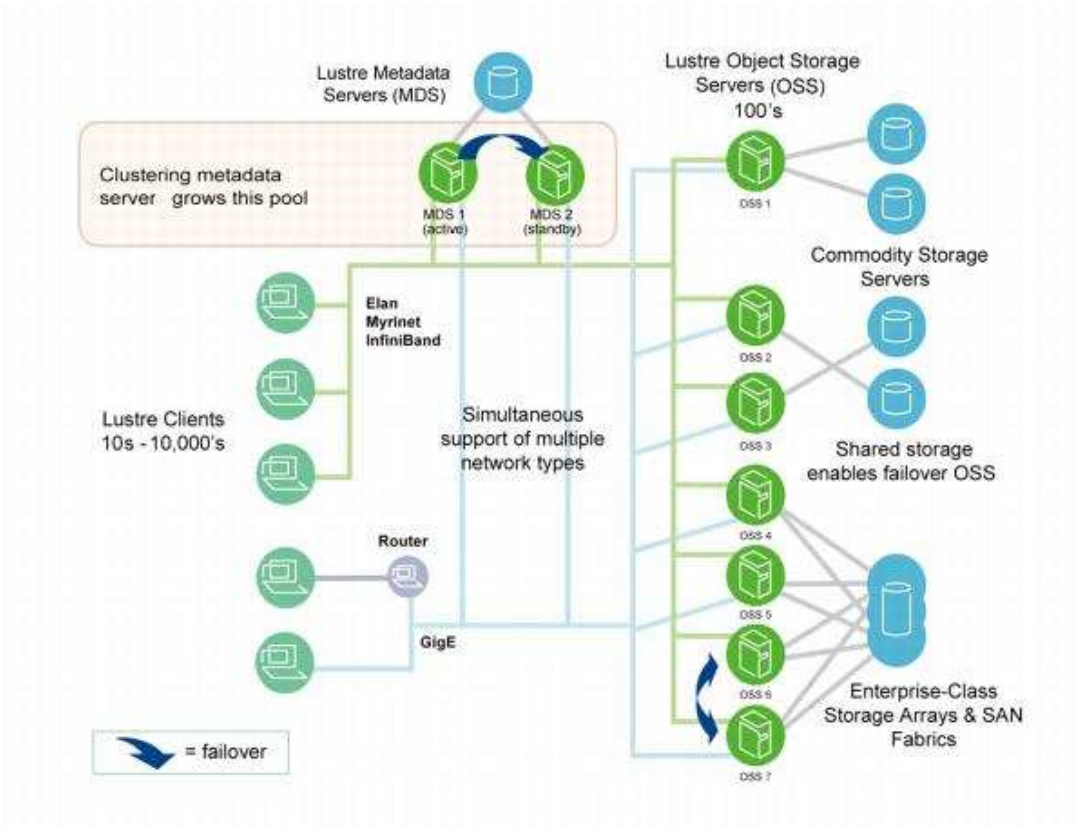

Figure 1 - An example of Lustre Cluster, source [16]

6.4. Lustre installation

The installation of Lustre was performed as described in the Installation Report [7]. The main steps for all the nodes (including clients) are:

i. Install the RPMs
ii. Modify the boot menu to boot on the new Lustre kernel, and reboot the node
iii. Check that the **`/lib/modules/`*`version`*`/kernel/net/lustre`** directory hosts the 4 following modules:
   - lnet.ko
   - libcfs.ko
   - ksocklnd.ko
   - lnet_selftest.ko
iv. Add the **`lnet`** options to the **`/etc/modprobe.conf`** file
v. Load the **`lnet`** module
vi. Configure and start up the Lustre network
vii. ***On the servers only***, format the disk with the **`mkfs.lustre`** command
viii. ***On all the nodes***, mount the partition with the **`mount.lustre`** command

7. Platform initial configuration

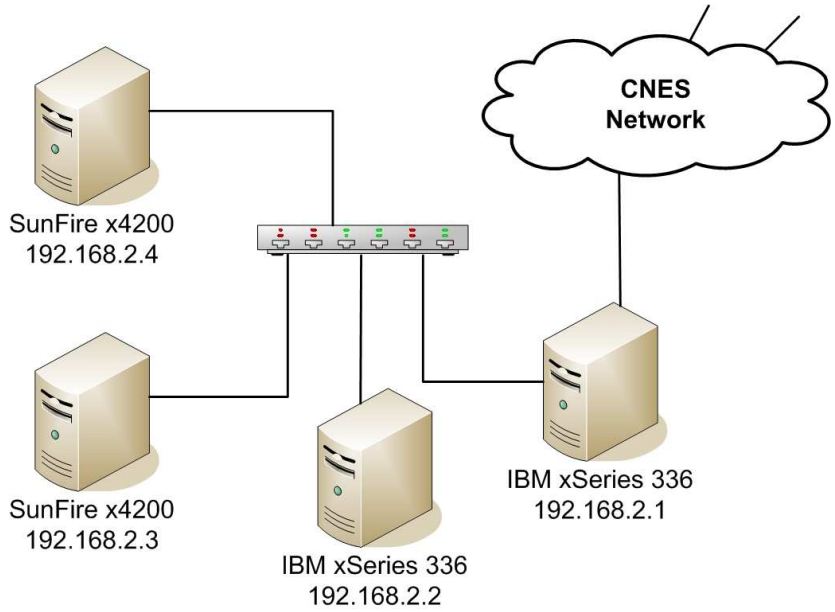

Figure 2 – Overview of the test platform

As we can see on Figure 2, the test platform consists of 4 servers, two of which are SunFire X4200, while the two others are IBM xSeries 336. They are all connected together through a 100 Mbps Ethernet network, on the **`192.168.2.0`** subnet.

One of the two IBM servers is furthermore connected to the **`CNES Network`** subnet, so that the system can be reached from outside, and all the nodes are configured so that they can communicate by SSH without any password.

The nodes are all equipped with 64 bits x86 processors, and the Linux version installed on is Red Hat Enterprise Linux 5 (RHEL5) [1], running with the version 2.6.18-8 of the Linux kernel.

Table 1 shows a short overview of the servers characteristics.

Server	IBM xSeries 336	SunFire X4200
CPU	2 * Intel(R) Xeon(TM) Dual Core	AMD Opteron(TM) Processor 254 Dual Core
CPU frequency	3200 MHz	2792 Mhz
Cache size	2048 KB	1024 KB
Memory	4GB (4049812 kB)	4GB (3995568 kB)

Table 1 - Servers characteristics

The two IBM servers are basically considered, and thus configured, as Lustre clients, while the two SunFire are configured as Lustre servers. The default configuration of the Lustre servers is one MGS/MDT server, and one OST.

8. Initial bottlenecks and platform improvements

8.1. Bandwidth improvement

Some basic tests were performed first, to search for the potential Lustre system bottlenecks. As the original network was only 100 Mbps (the Gigabit Ethernet router wasn't yet available), we guessed this would be the first bottleneck. So we measured the time spent to copy files, on a client, from a local ext3 partition to the Lustre one, and then in the other direction, and compared it to the time needed for a local copy (between two local ext3 partitions).

The results of the first test are given in Table 2. As we can see, between 85 and 90% of the bandwidth is used, which means that it is clearly saturated. So we performed the same test later with the Gigabit Ethernet router, and found the results exposed in Table 3.

Data	Way	Time	Throughput (MB/s)	Bandwidth utilization
1 file of 2866 MB	Ext3 → Lustre	4m. 13.091s	11.304	90.60%
	Lustre → Ext3	4m. 15.699s	11.208	89.66%
	Ext3 → Ext3	1m. 15.898s	37.760	

Table 2 - Time to copy files on the client, with a 100Mbps bandwidth

Data	Way	Time	Throughput (MB/s)	Bandwidth utilization
1 file of 3 000 MB	Ext3 → Lustre	1m. 15.720s	39.620	31.70%
	Lustre → Ext3	58.290s	51.467	41.17%

Table 3 - Time to copy files on the client, through a Gigabit Ethernet network

Now the bandwidth utilization does not exceed 40%, so the bottleneck is not anymore the network, but probably the hard disk, since writing on the disk is an operation expensive in time. The solution chosen to solve this is to attach a Storage Area Network (SAN) to the servers.

8.2. Storage Area Network

8.2.1. Introduction

Basic storage devices are called Direct Attached Storage (DAS), since the storage device (typically a hard disk) is directly attached to the server, and thus they are not easily scalable. On the contrary, there exists two others storage management system, that are much more scalable.

The first one is the Network Attached Storage (NAS), in which the storage device and the file system are remote, and connected to the server and its application layer through a network. In this case, data are accessed by file. The second one is the Storage Area Network (SAN), in which the file system comes among the application layer, and only the storage device is remote. There, data are accessed by block. These three systems are explained in Figure 3.

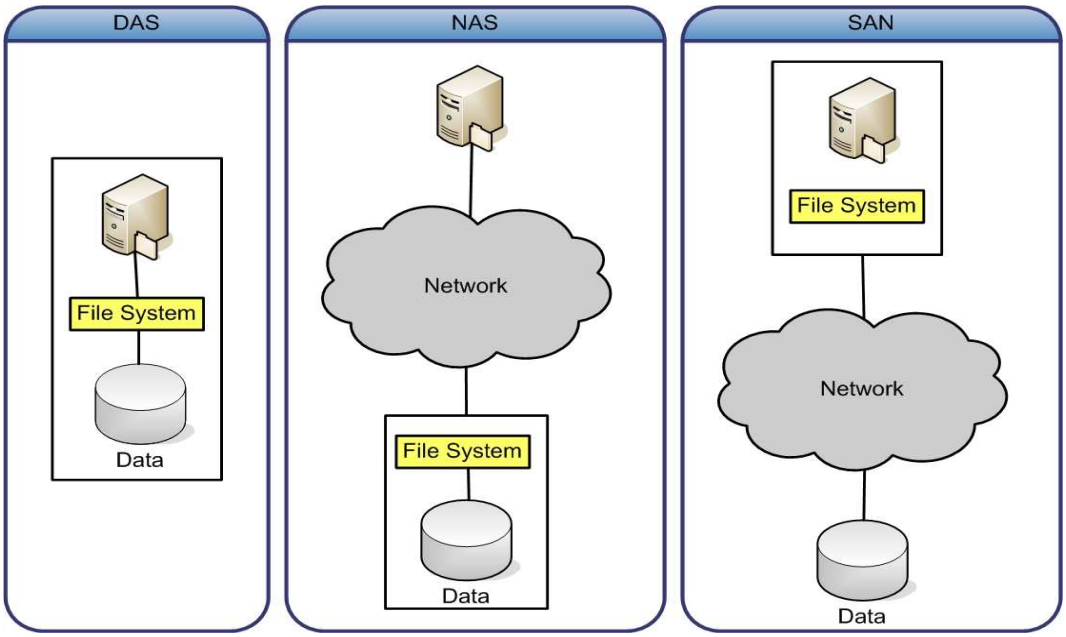

Figure 3 – Overview of DAS, NAS and SAN

The main advantage of both NAS and SAN is their scalability, since several storages devices can be added or removed easily, simply by plugging or unplugging it to the network, and configuring the server.

On the previous tests, we were using a DAS. But as we have seen, its performance limits have been reached, so we decided to create a SAN. Our new configuration is explained on the Figure 4. Notice that the SAN components are connected together through a technology called Fiber Channel (orange wires in the figure).

You can find in the Table 4 an overview of the SAN terminology, and equivalent well known technologies.

SAN	Equivalent technology
Fiber Channel	Ethernet
HBA Card (Host Bus Adapter)	Ethernet Card
World Wide Name (WWN)	Mac address
Small Computer System Interface (SCSI)	TCP

Table 4 - SAN terminology

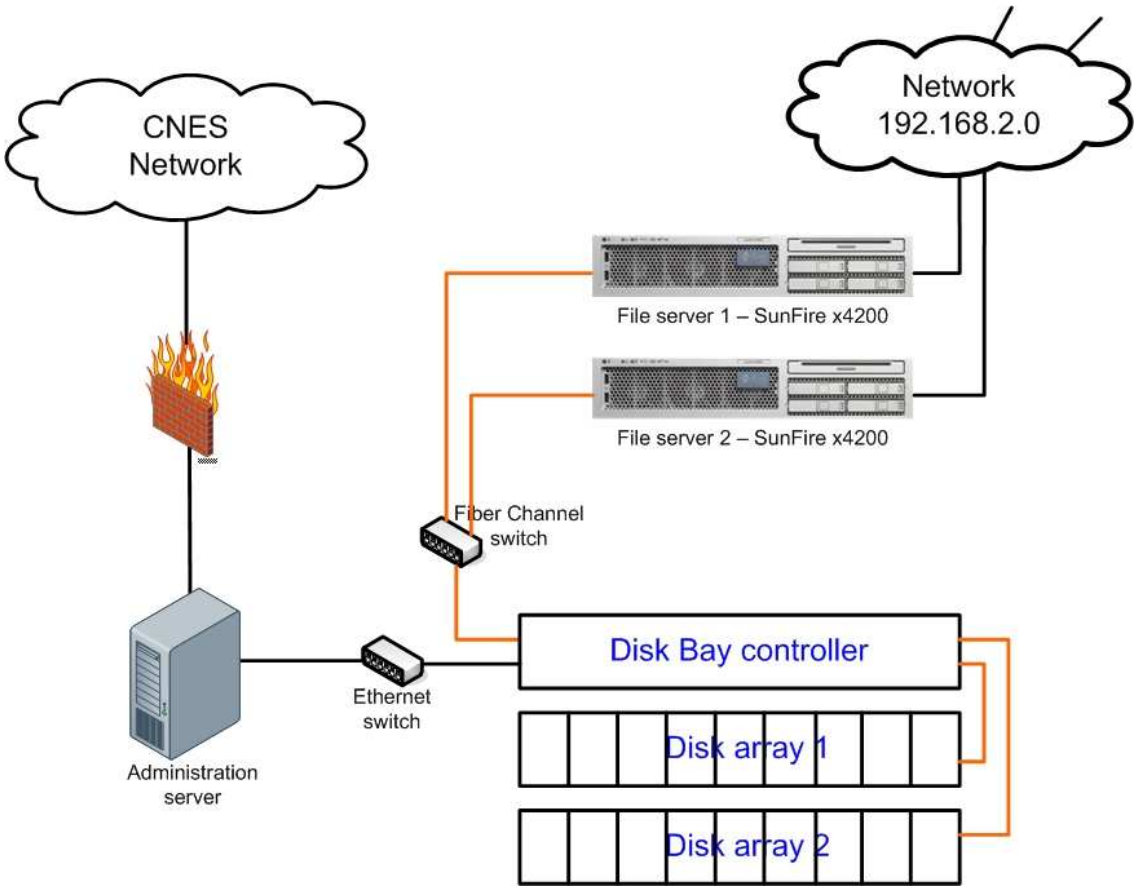

Figure 4 - The SAN hardware configuration on the test platform

8.2.2. Installation

The SAN installation and configuration required the hardware installation, which consisted of connecting the servers and the disk bay through Fiber Channel router and wires. This configuration allows a throughput up to 2 Gigabits per second.

Once the hardware connection done, we had to set up the bay controller: delete all existing configurations, create virtual disks, and then create LUNs – Logical Unit Numbers. At last, we mapped the LUNs to the servers. The resulting configuration is exposed in Figure 5.

Meanwhile, we installed the HBA Card driver on the 2 servers connected to the SAN. HBA Cards allow Fiber Channel connections. Unfortunately this installation could not be performed on the Lustre kernel, and installing it on the original kernel and then reboot on the Lustre one did not solve the problem. So the solution was:

- ★ uninstall Lustre, and reboot on the original kernel
- ★ install the driver[2]
- ★ re-install Lustre
- ★ reboot on the Lustre kernel

[2] The driver sources and the documentation can be found on the Qlogic website [8].

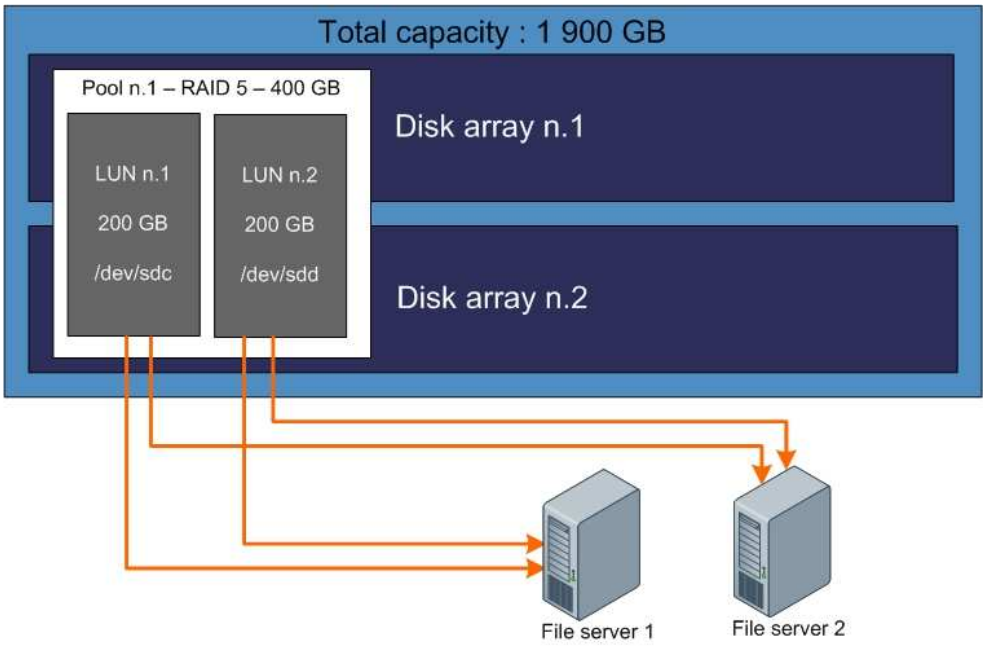

Figure 5 – Test platform SAN architecture

8.2.3. Performances

This time, we could not make copies from a file, since it would have used the client disk, what we did not want. Instead, we created files using the '**dd**' command[3]. This test was performed both locally and remotely, and the results are exposed in Table 5. In the first case, the OST Lustre server was itself configured as a Lustre client too, so that performing the tests on this special client would not be network dependant.

Data	Client	Time	Throughput (MB/s)	Bandwidth utilization
1 file of 3 000 MB	`Remote`	`31.523s`	`95.168`	`76.13%`
	`Local`	`28.271s`	`106.116`	`-`

Table 5 - Time for local and remote file copy on a SAN, with Lustre

As we can see, the bottleneck is the network again, since the local test is significantly faster than the remote one, for which the bandwidth can be considered as saturated. We will further on perform more tests to highlight this.

8.3. Comparing Lustre kernel and "original" Linux kernel

Because installing Lustre creates a modified kernel, we can wonder if performing ext3 or NFS tests on this kernel will not decrease their efficiency. So, to compare the efficiency of both kernels, we perform Iozone test[4]. When running remote tests, the server kernel was always the same as the client one.

[3] See Appendix I for details on the commands used.
[4] See Appendix I again.

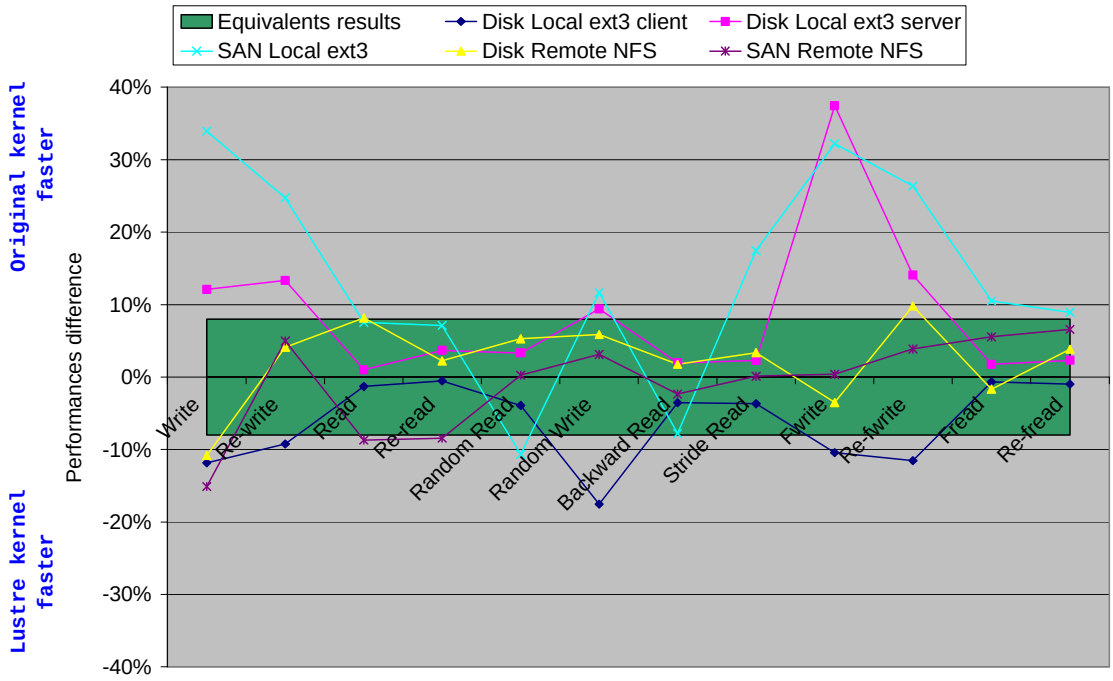

Figure 6 - Differences of efficiency between the 2 kernels

We can see in Figure 6 that performances are globally as fast for both kernel. There are however some exceptions to this:

- Sequential writing on ext3 on the server are performed significantly faster by the original kernel – both on local disk and through the SAN.
- On the contrary, sequential, and random, writeing on ext3 on the client are performed faster by the Lustre kernel.

Because performances are equivalent for remote operations, and because remote operations are the main interest of our system, the following experiences will all be driven on the Lustre kernel. Thus, all the experimentations can be driven on the same kernel, and tedious reboots are avoided.

8.4. A mix of disk and SAN for Lustre data nodes

As Lustre allows it, we started two OSTs on the same server, one using the internal disk, and the other one using the SAN. Then, performing '**dd**' tests, we checked on which OST the data had been saved, executing the '**df**' command on the OST server. We realized that the data were stored only on one of both OST, and the performance was exactly the same as if only this OST – the one where the data had been stored – was started.

Lustre does not write automatically on the 2 devices, resulting in lower performances, but thus guaranteeing a bigger availability. Indeed, if data are written on 2 different OST, the fall of any of the 2 servers would imply the loss of these data.

So we didn't include tests with a mix of disk and SAN in the following experimentations, because they are not that interesting in terms of performances.

9. Comparison tests

9.1. Cache efficiency

9.1.1. Iozone and the caches effects

Iozone allows a three dimensional view of one file system performances, on which we can observe the different cache levels.

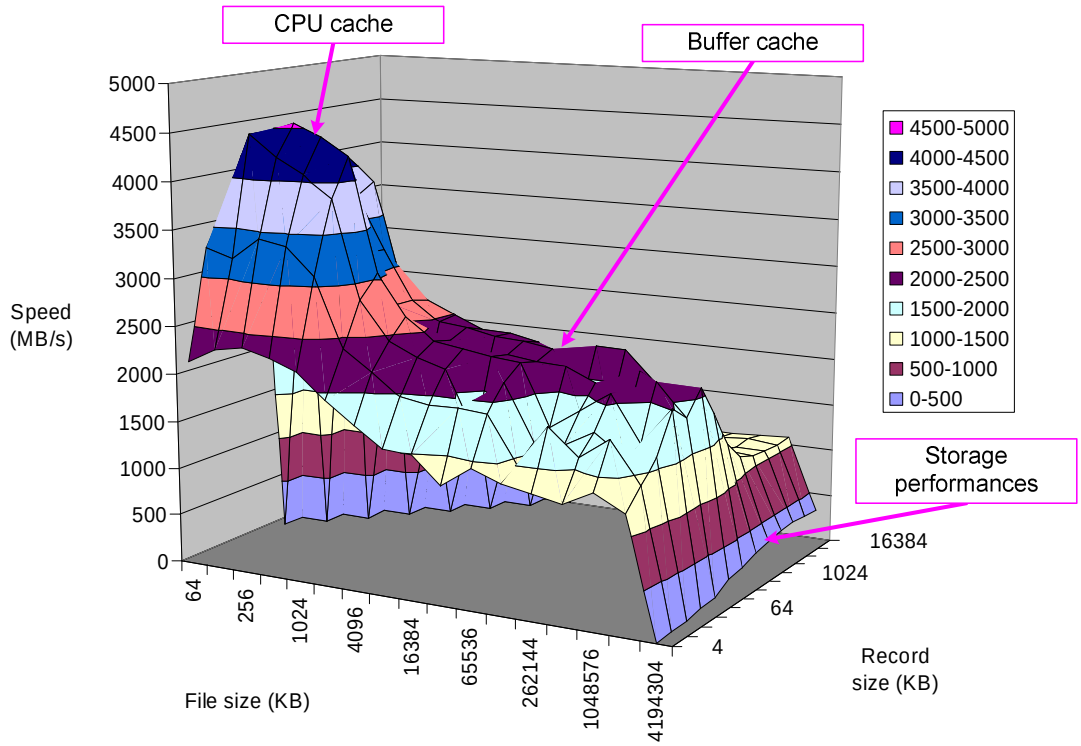

Figure 7 - Caches effect on ext3 – Re-read report

As we can see on Figure 7, when all the data can fit in the CPU cache (up to 1MB), we get a very high efficiency, around 4 GB/s. Larger files do not fit in this CPU cache; however, up to 2 GB, they fit in the buffer cache, which is a little bit slower, but can give a speed around 2 GB/s. Files larger than 2 GB do not fit in any cache, and so we can measure the storage efficiency.

In this document, all occurrences of the “cache” word refer to the buffer cache, unless it is clearly specified.

9.1.2. Comparing Lustre to ext3, ext2, and NFS

We first executed the Iozone test on files smaller than 2 GB (memory size is 4 GB on each machine), performing the default operations, and printed the mean throughput for each one. On Figure 8 and Figure 9, we just selected the operations in which the buffer helped.

As we can see on Figure 8, the file systems taking most advantage of the cache is ext3. It is up to twice faster than the two others for reading. Lustre and ext2 have similar results.

Remotely, NFS over ext3, is once again more efficient than Lustre, even though NFS seems to lower a little bit ext3 performances (see Figure 9). Lustre remote performances are really similar to local ones. This is quite normal, because we just added the network layer – that does not interfere in the cache efficiency – to the system.

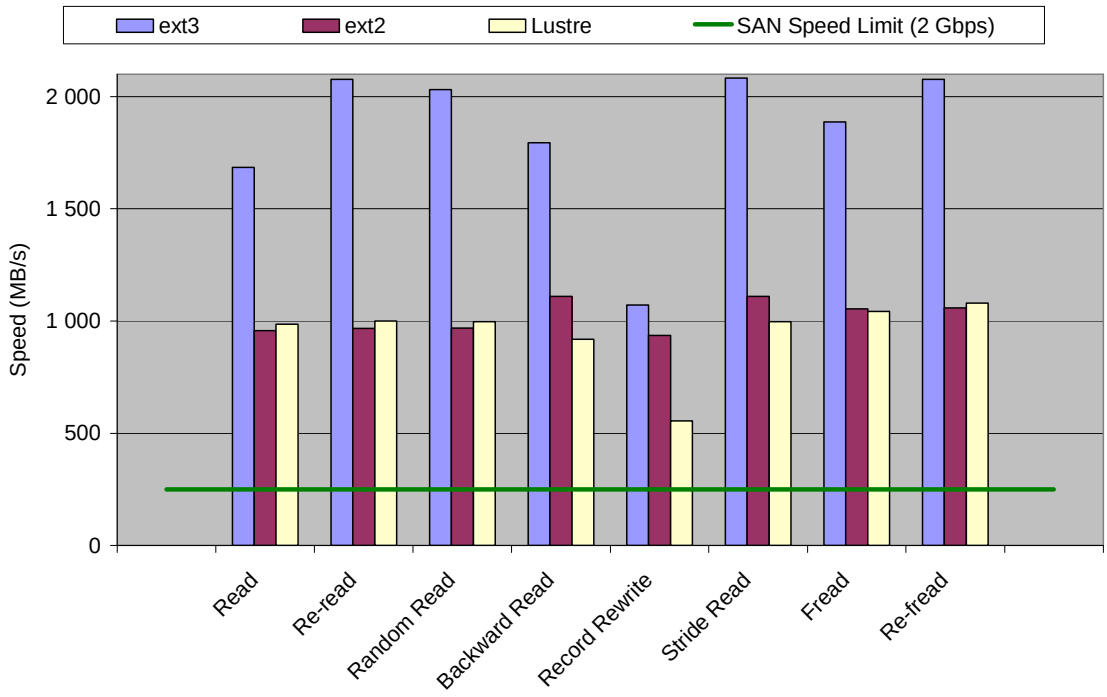

Figure 8 - Local cache speed

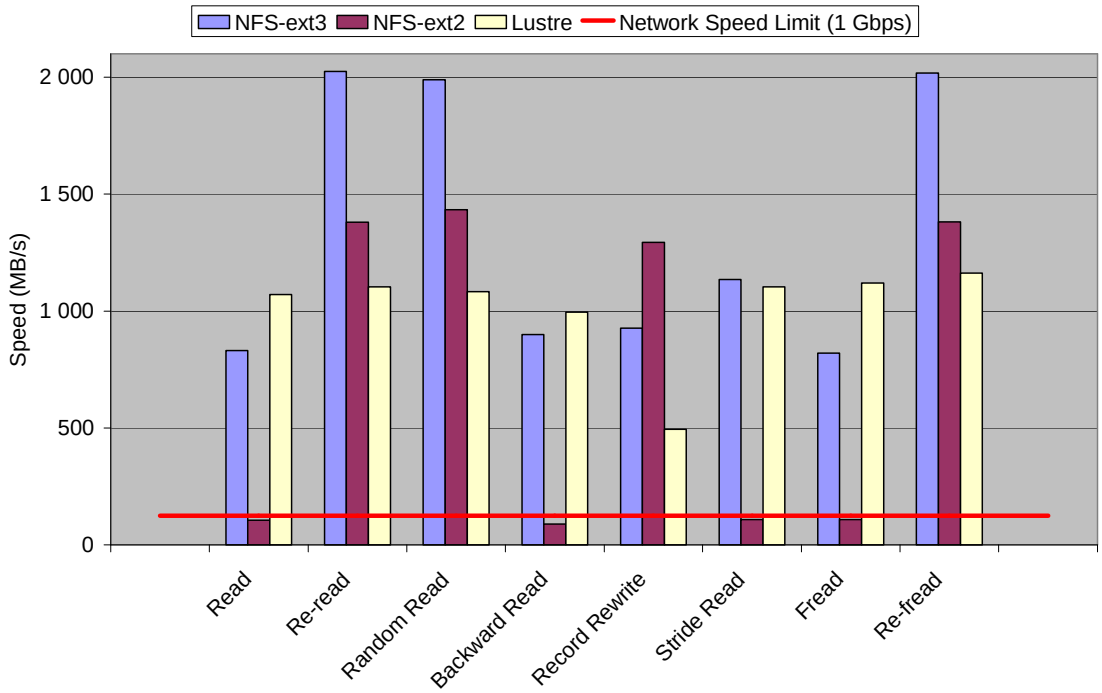

Figure 9 - Remote cache speed

On the contrary, NFS slows down ext2 a lot. Indeed, 4 operations (read, backward read, stride read and fread) do not seem to take advantage of the cache. The 4 other operations are performed faster than locally, perhaps because the clients buffer is faster than the servers one.

9.2. Local throughput

The first test we performed was writing and reading through the '**dd**' command, and then we performed Iozone tests with an 8 GB file, and 1 MB blocks[5].

9.2.1. The ext2 'pre-empted commit'

Running some quick tests, we got some strange results: we expected reading to be faster than writing, which never happens with ext2 and ext3, whatever the configuration is. To check this point, we measured the throughput on the Fiber Channel network, between the server and the SAN disks, while writing on the partition, and then while reading it.

As we can see on Figure 10, data are sent at an average throughput of around 110 MB/s. The reason why writing looks so fast to the user is that the server commits the operation before it has really ended. We can guess that, once all data are loaded in memory, the system can assure all data are going to be written safely, and the process returns, whereas around 1,5 GB (server memory is 4 GB) of data have not been written yet.

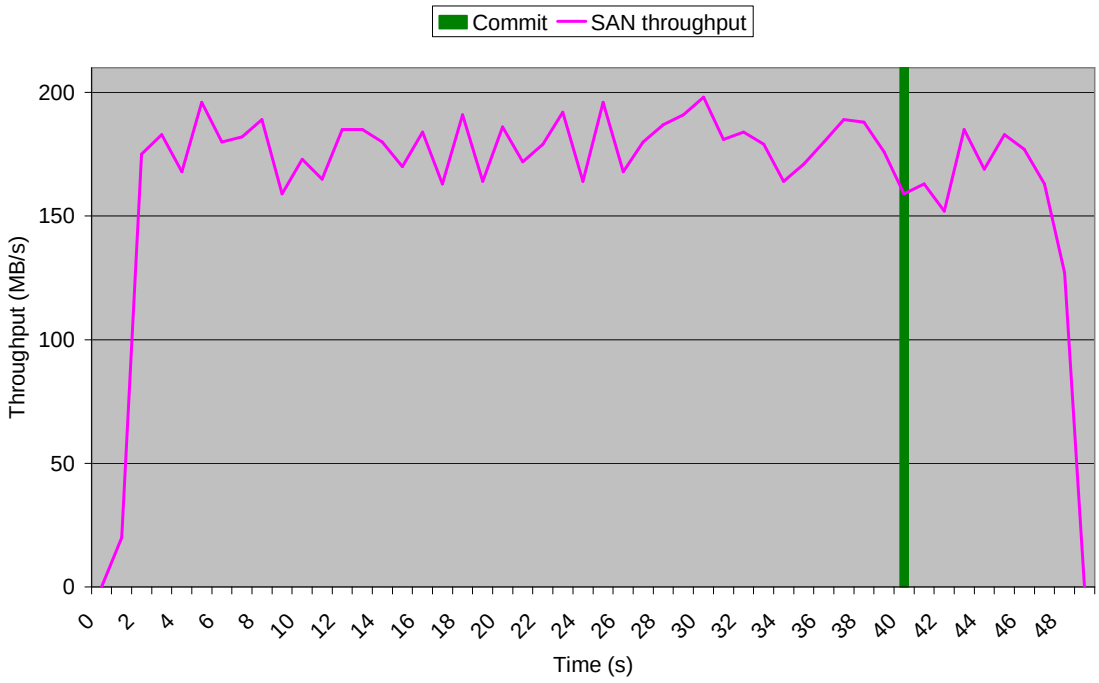

Figure 10 - Fiber Channel network throughput while writing on ext2

We have the same 'pre-empted commit' with ext3, which is based on ext2. On the contrary, Lustre does not return before all data have been written on the storage device (Figure 11).

[5] See Appendix I

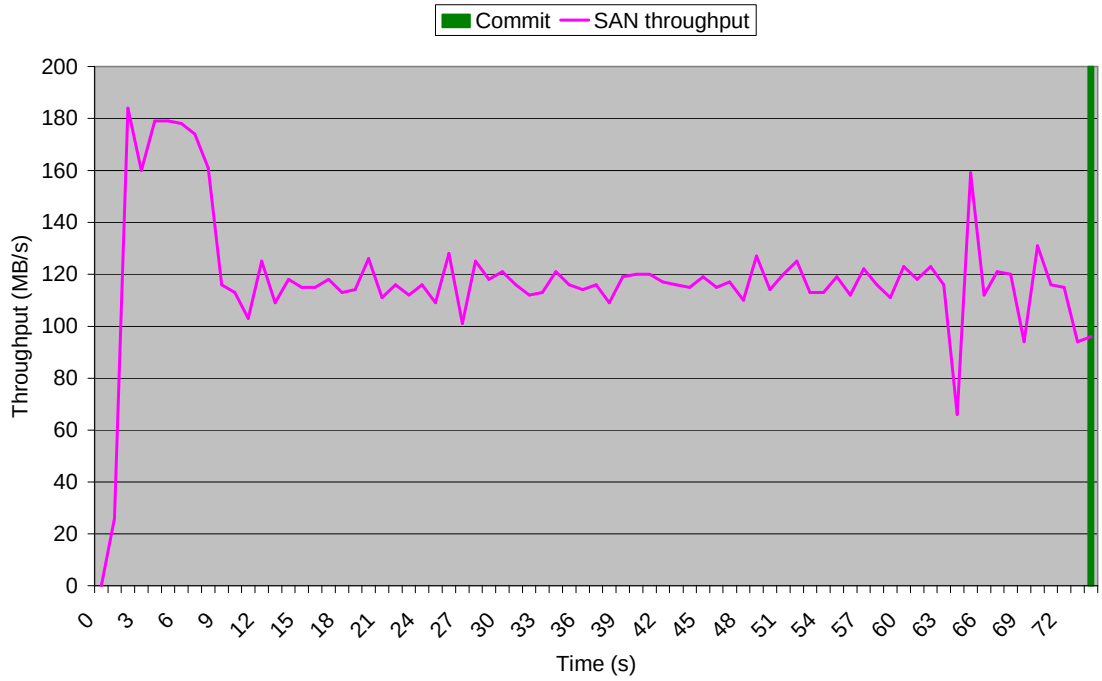

Figure 11 - Fiber Channel network throughput while writing on Lustre

9.2.2. Results

As we explained before (see 9.2.1), ext2 and ext3 seem to be very fast for writing, because of the 'pre-empted commit' effect., as we can see on Figure 12.

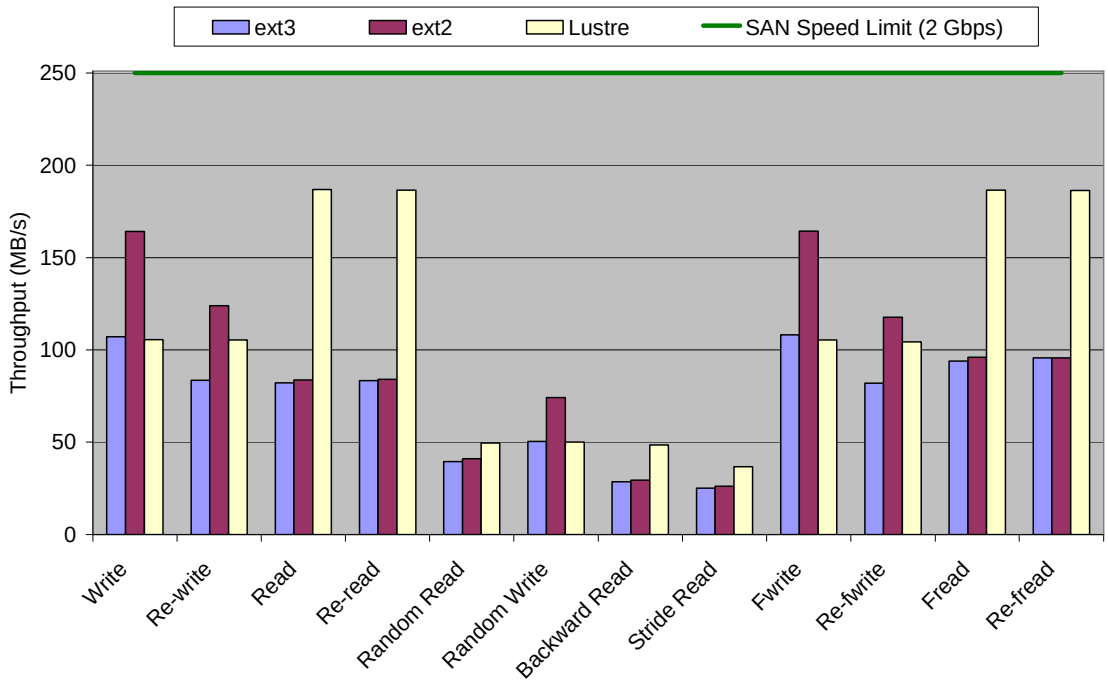

Figure 12 - Local throughput

We expected the ext2 and ext3 results to be better than Lustre ones, since Lustre is an additional layer to ext3. However, only ext2 is faster than Lustre, and only for writing, thanks to the 'pre-empted commit' effect. For reading, Lustre is much faster than the two other file systems.

Since ext3 is an extension of ext2, adding a journal, writing is quite slower, because there is more data to be written. However, that does not change anything for reading, since the journal is not modified by this operation – and thus performances are equivalent when reading on ext2 or ext3.

At last, we can see that in both experiences, reading with Lustre on the SAN reaches a throughput of almost 200 MB/s, taking into account the latency due to the communication between the OST and the MGS/MDT servers, which lowers this throughput. Since the Fiber Channel network has an upper theoretical limit of 2 Gb/s, or 250 MB/s, this network has an utilization of 80%. So reading on the SAN with Lustre saturates the Fiber Channel network.

9.2.3. Bottlenecks

As we have seen previously, using a SAN is in all our tests more efficient than using the server's internal disk. So in the case of using the internal disk, this device is the bottleneck of the system.

```
top - 13:30:39 up 22:09,  1 user,  load average: 8.99, 2.95, 1.13
Tasks: 403 total,   3 running, 400 sleeping,   0 stopped,   0 zombie
Cpu(s): 0.2%us, 25.4%sy, 0%ni, 0.0%id, 72.8%wa, 0%hi, 1.7%si, 0%st
Mem:  3995568k total, 2494184k used, 1501384k free,   11320k buffers
Swap: 4192924k total,     152k used, 4192772k free, 1943968k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7074 root      18   0 63280 1584 1496 R   45  0.0   0:08.00 dd
```

Listing 1 - 'top' result when writing on ext3

The bottleneck when writing on ext2 or ext3, over the SAN, cannot be the Fiber Channel network, since we measured a 110 MB/s throughput, and the network can support up to 250 MB/s. So we performed a '**top**' while writing on ext3, and found out that the bottleneck was the CPU, as we can see in Listing 1 (the same result has been observed on ext2).

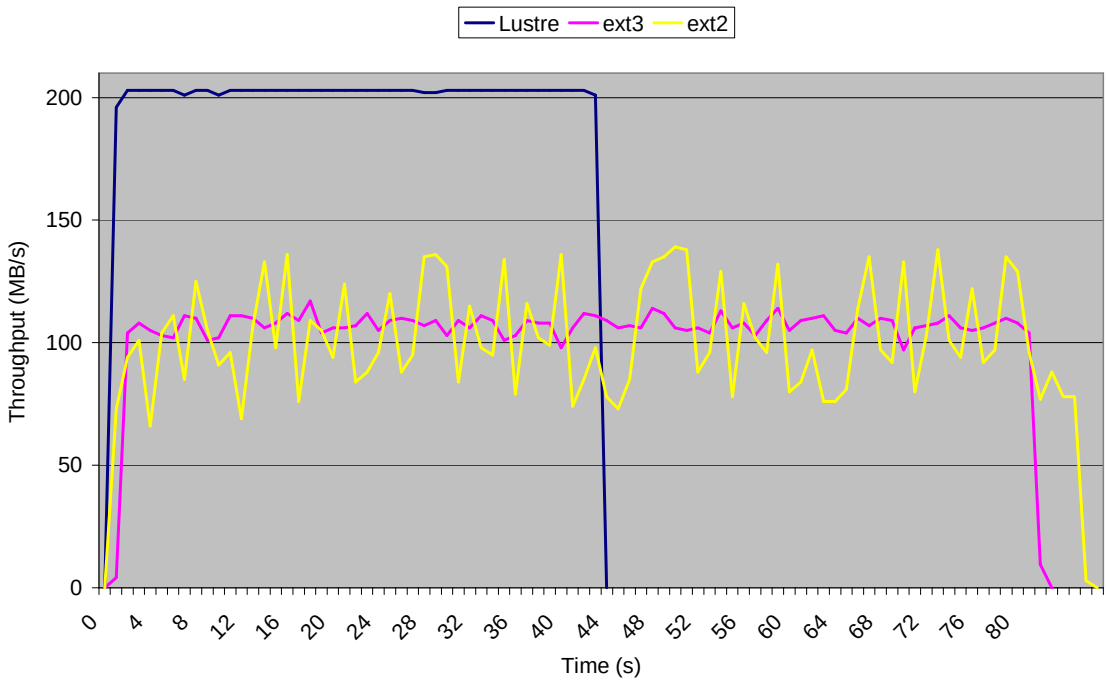

Figure 13 - Fiber Channel network throughput during local read

The process uses around 50% of the CPU, the idle time is null ('**`0.0%id`**'), and the CPU spends the great majority of its time waiting for I/O ('**`72.8%wa`**'). This intensive need of CPU is caused by the "pre-empted commit" effect.

On the contrary, when writing on Lustre, the bottleneck is not the CPU, what we can see with the '**`top`**' command. The process uses only 20 to 25% of the CPU, which is not saturated – more than 40% of idle time. We did not find out this operation bottleneck.

When reading on Lustre, as we can see on Figure 13, the Fiber Channel network throughput is upper than 200 MB/s, so the Fiber Channel network is, in this case, the bottleneck.

On the contrary, the average throughput when reading on ext3 is around 104 MB/s only, and the server CPU is not saturated. Once again, we have no idea of what the bottleneck could be for the ext2 and ext3 reads.

9.3. Remote throughput

9.3.1. Results with a Gigabit Ethernet network

As we can see on the Figure 14, Lustre is around twice faster than NFS. But what is very important here, and has to be noticed, is that reading (and probably writing) with Lustre on the SAN is only limited by the network.

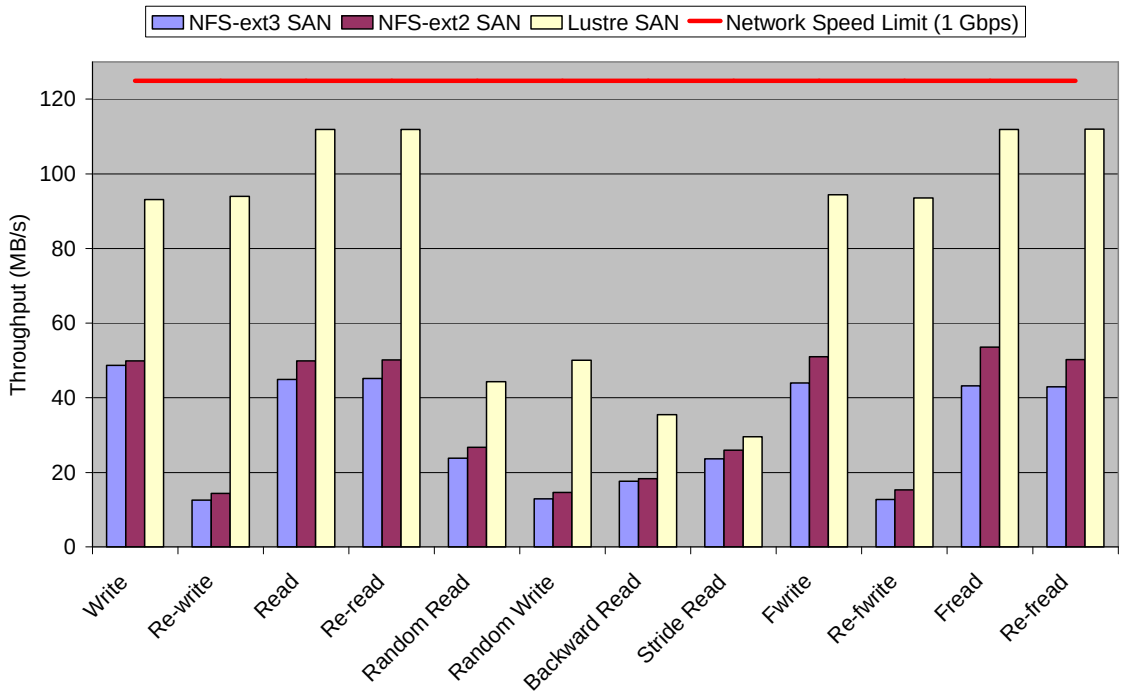

Figure 14 - Remote throughput

We can guess that when using Lustre, sequential reading and writing are limited by the network, since their performances are worse than local ones. But for non-sequential operations, which are slower, the throughput remained the same, as expected.

On the contrary, in the case of NFS, not only the network has been added to ext2 and ext3, but also the NFS overhead, which makes performances worse than local ones. At last, we can notice that NFS transfers the 'pre-empted commit' effect, since re-writing is still slower than writing.

9.3.2. Network bonding

Since the network seemed to be the bottleneck is some cases, we decided to add a Gigabit Ethernet connexion to one client and one server (basically, the Lustre OST), and to perform bonding between the two connections, so we can double the theoretical network throughput. The results are given in Figure 15.

As expected, NFS performances were exactly the same as when using a single Gigabit Ethernet connexion (NFS performances were not limited by the network). And as expected, Lustre performances that were limited by the network are increased, even though they do not reach the level of performance that was achieved locally. However, it confirms one of the previous hypothesis: the Gigabit Ethernet network is a bottleneck for Lustre operations.

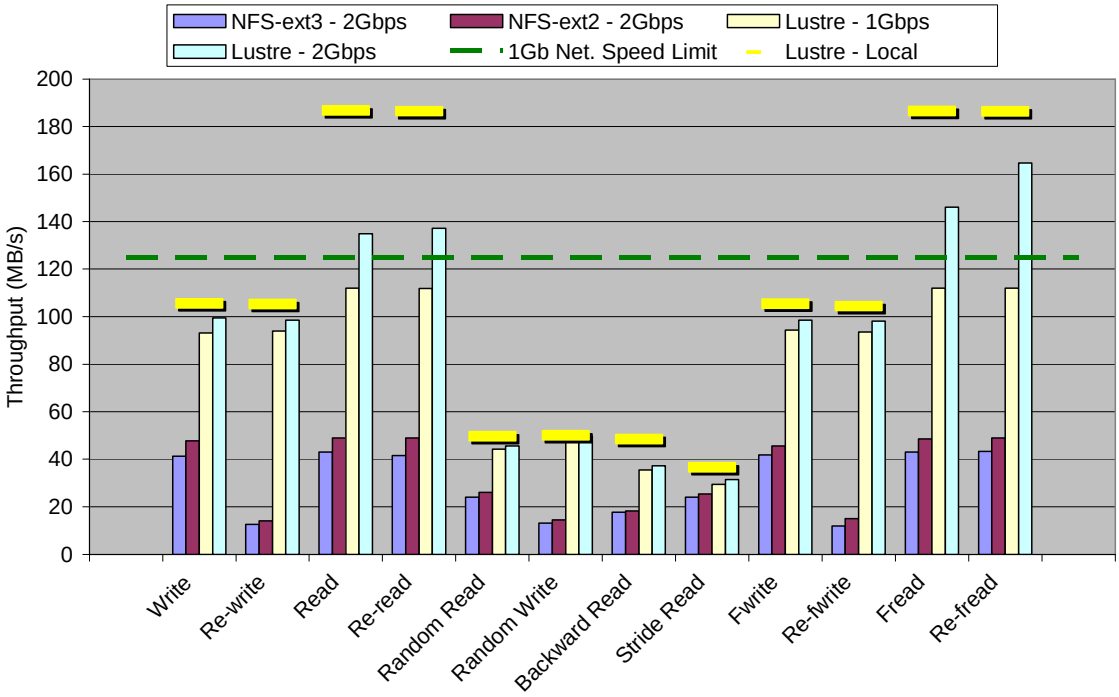

Figure 15 – Bonding network: comparison to local performances

9.3.3. Comparing Lustre, NFS and FTP

FTP is a very efficient file transfer protocol, using the maximum of the available bandwidth, as we can see on Listing 2, performed through the network with bonding:

```
ftp> put "|dd if=/dev/zero bs=1024k count=8192" /dev/null
local: |dd if=/dev/zero bs=1024k count=8192 remote: /dev/null
227 Entering Passive Mode (192,168,2,4,48,176)
150 Ok to send data.
8192+0 enregistrements lus
8192+0 enregistrements écrits
8589934592 octets (8,6 GB) copiés, 47,851 seconde, 180 MB/s
226 File receive OK.
8589934592 bytes sent in 48 seconds (1.8e+05 Kbytes/s)
```

Listing 2 - Writing a file through FTP

The throughput is somewhat inferior to the theoretical upper limit, probably because we do not use a real 2 Gbps network, but bonding, that may lead to some overhead.

Despite that network efficiency, FTP writings are much slower than local writings on ext2. They remain however more than twice faster than NFS ones. On the contrary, Lustre writings, which were much slower locally (45% slower) are now almost as fast as FTP ones (12% slower).

Local readings were faster on Lustre than on ext2, and the difference grows a little bit: Lustre was 43% faster locally; it is remotely 49% faster than FTP and 56% faster than NFS.

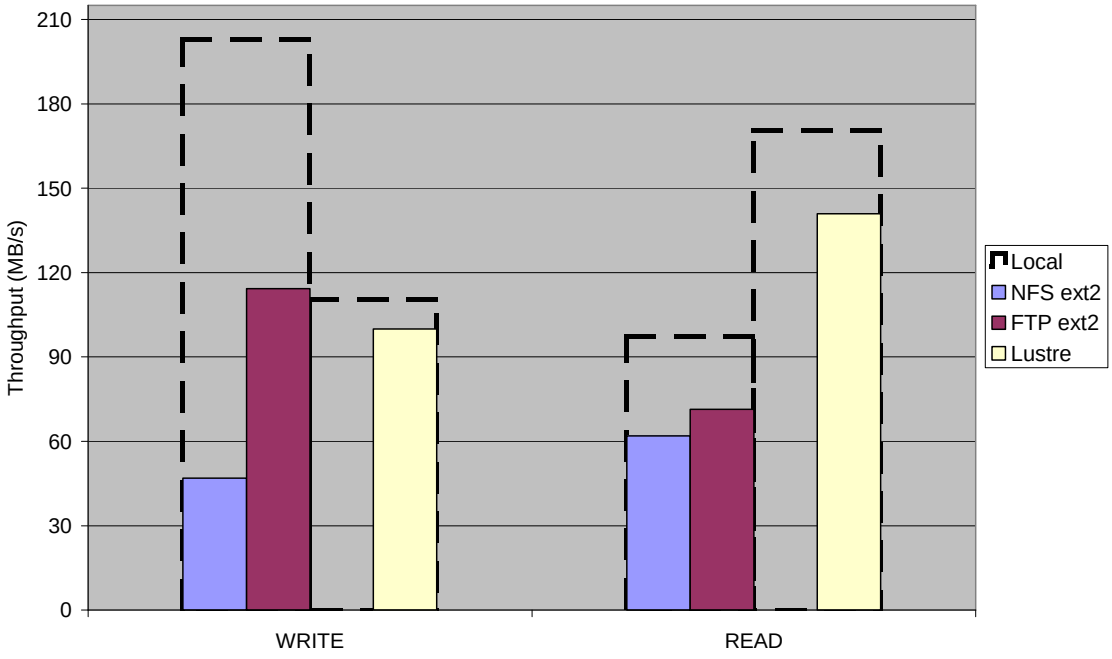

Figure 16 - FTP, NFS and Lustre throughput performances

The thing is that for Lustre, the overhead (around 10% for each operation) is caused only by the network; whereas both NFS and FTP add an additional software layer that leads to another additional cost. In the case of FTP, this software overhead is very important, as we con see on Listing 3. Moreover, FTP adds a communication latency, which may reach in some cases 5 seconds.

```
top - 15:43:24 up 1 day, 22:43, 1 user, load average: 0.86, 0.63, 0.42
Tasks: 408 total,   2 running, 406 sleeping,   0 stopped,   0 zombie
Cpu(s): 0.2%us, 17.6%sy, 0%ni, 55.9%id, 0%wa, 1.3%hi, 25.0%si,  0%st
Mem:  3995568k total, 2356668k used, 1638900k free,   14764k buffers
Swap: 4192924k total,     280k used, 4192644k free, 2012244k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
11616 root      15   0 39596 1032  508 R   52  0.0   0:07.60 vsftpd
```

Listing 3 - 'top' result during an FTP transfer

As a conclusion, Lustre seems to be a really better choice than either FTP or NFS. It is not only much faster than the two others for reading, but it is also almost as fast as FTP for writing, and much faster than NFS. Most of all, it provides an easy way to manage remote files – much easier than FTP – and it is a very scalable system.

9.3.4. Overhead

To compare the Ethernet overhead generated by Lustre, NFS and FTP on the network, we measured the amount of data transmitted while writing or reading an 8 GB file, through the 3 different protocols. We performed both NFS and FTP over ext2.

Data in MB/s		Overhead transmitted	Overhead received	Total overhead	
Lustre	**Write**	364,32	479,44	843,75	**10,30%**
	Read	478,65	296,01	774,66	**9,46%**
NFS-ext2	**Write**	438,99	487,89	926,89	**11,31%**
	Read	474,15	412,63	886,78	**10,82%**
FTP-ext2	**Write**	396,86	478,95	875,81	**10,69%**
	Read	450,01	444,68	894,69	**10,92%**

Table 6 – Lustre, NFS and FTP Ethernet overhead for an 8GB file

As we can see, the overhead generated by NFS and FTP is a little bit higher than the one generated by Lustre. It has to be noticed that, as expected, this Ethernet overhead seems to cover single-handedly the throughput difference between local and remote uses of Lustre, observed in the previous paragraph.

The measures were made for one 8GB file. A test with 1024 files of 8MB gave very similar results for Lustre (around 10% of overhead) and NFS (around 11%). This test with quite small files was not performed with FTP, because we did not know how to perform it.

When transferring an 8 GB file on the Fiber Channel, there are around 8,5GB of data passing on this network, which means an overhead between 6 and 7% of the transferred data amount. It is more or less the same values as observed on the IP network.

9.3.5. Behaviour depending on the file sizes

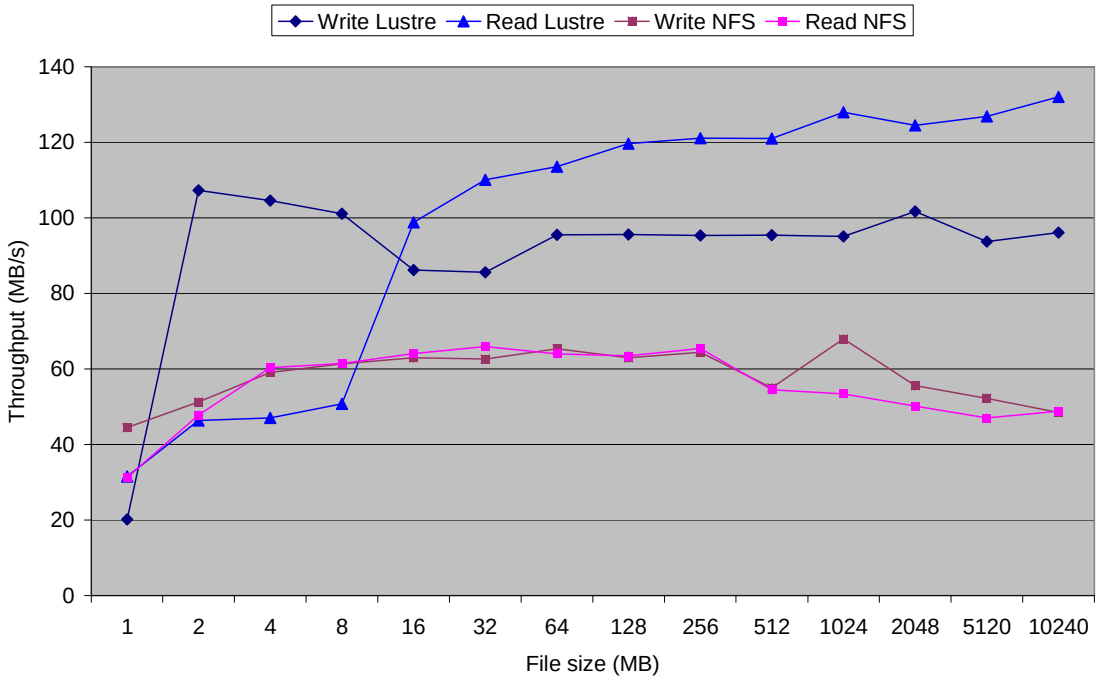

Figure 17 - Remote throughput depending on the file size

For this test, we created as many files as necessary to manage 10 GB of data. That means that the number of files used is inversely proportional to the files size.

As we can see in Figure 17, it is not recommended to use Lustre for files smaller than or as big as 8 GB, since reading this small files is faster with NFS. However, for bigger files, Lustre is obviously faster, and the reading speed increases with the file size, whereas it seems to decrease with NFS.

Notice that the very slow result when writing a 1 MB file with Lustre comes along with an incredibly high overhead: more than 60% ! On the contrary, for every other writings, the overhead stands between 9,3% and 10,8% for Lustre, and between 11,1% and 11,8% on NFS, no matter the file size and the number of files.

Overheads are similar for both file systems for readings: between 10,1% and 11,7%, NFS being a little bit higher – 11,14% in the mean – than Lustre – 10,90%.

9.4. Access time results

9.4.1. Local performances

We can notice that ext2 and ext3 are one order of magnitude faster than Lustre, for both touch and remove tests – up to 55 times faster.

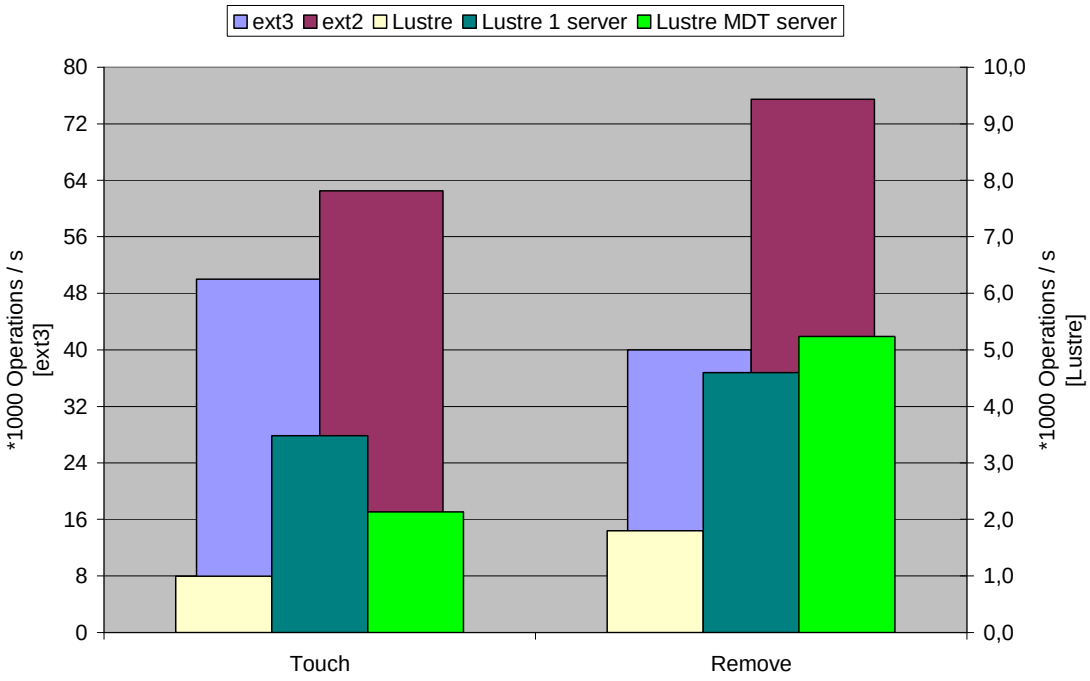

Figure 18 – Local access time efficiency (the higher, the better)

Such a difference can be explained by the fact that creating or removing a file from Lustre implies TCP/IP communication between the metadata server and the OST.

To confirm this hypothesis we performed the test on the MDT server, an then on a modified platform, where Lustre MGS/MDT and OST servers are installed on the same physical machine. In both case, part of the TCP/IP communication cost should be avoided. Results are shown in Figure 18.

As expected, the throughput is higher when running the tests on the MDT server – at least twice faster for 'touch', and 3 times for 'remove'. This confirms that the cost of the TCP/IP communication between the servers is non-negligible, even though the biggest part of the difference between ext3 and Lustre must be imputed to the Lustre overhead.

9.4.2. Remote performances

Remote Lustre performances are equivalent to local ones, which means that TCP/IP communication between client and server has almost no impact on the access time.

On the contrary, as we can see in Figure 19, exporting ext2 or ext3 through NFS adds such a cost that NFS is even slower than Lustre – whereas locally, ext2 and ext3 were up to 55 times faster. In fact, NFS performances are 60 to 70 times slower than ext2 and ext3.

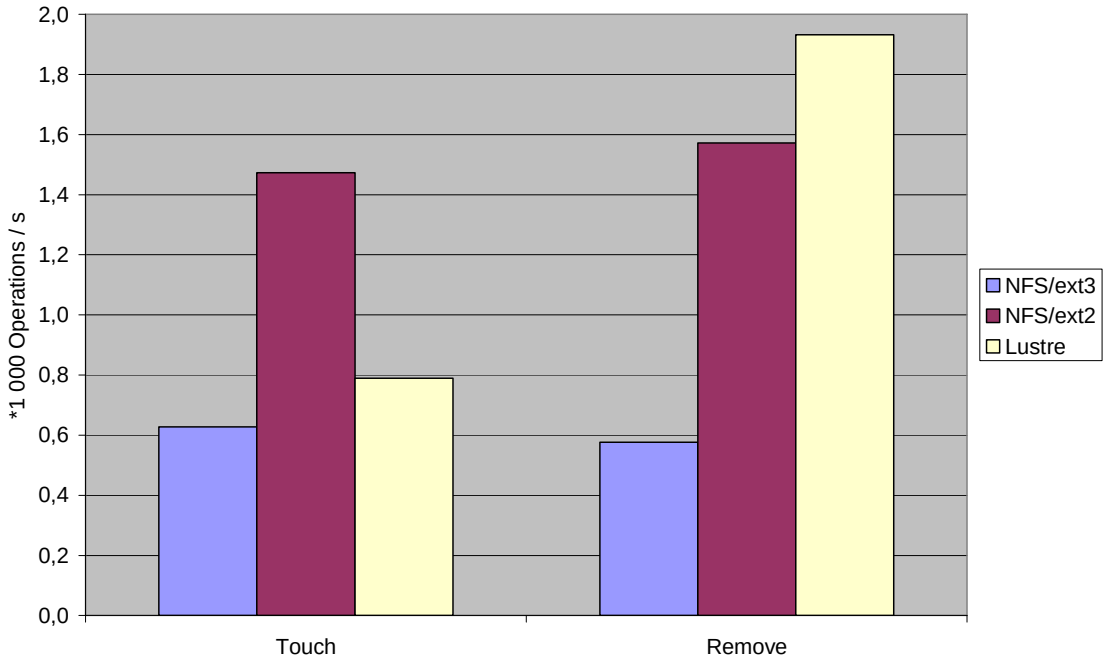

Figure 19 - Remote access time efficiency (the higher, the better)

9.5. Parallel tasks

We first performed tests using the Iozone functionality. Since the results we got were a little bit strange, we decided to perform other tests with the '**dd**' command. We performed these tests by running several parallel tasks (write or read) on the same client, and then on 2 clients. We tested 2 different behaviours: all the tasks trying to access to the same file, and one file for each task.

Because we performed these tests quite late in the planning, we did not have the time to perform the tests with two clients, and thus we only present in this report the result we got with a single client.

It is to notice that it is almost impossible to run various tasks trying to write concurrently on the same file through NFS: the processes seem to be locked.

Whether all the processes write on the same file or not, the total throughput is almost constant when writing on Lustre, between 95 and 110 MB/s. It is not exactly constant: it decreases a little bit, because of the load of the server. We observed the same phenomenon when writing on various files through NFS, but at a lower throughput – between 40 and 50 MB/s.

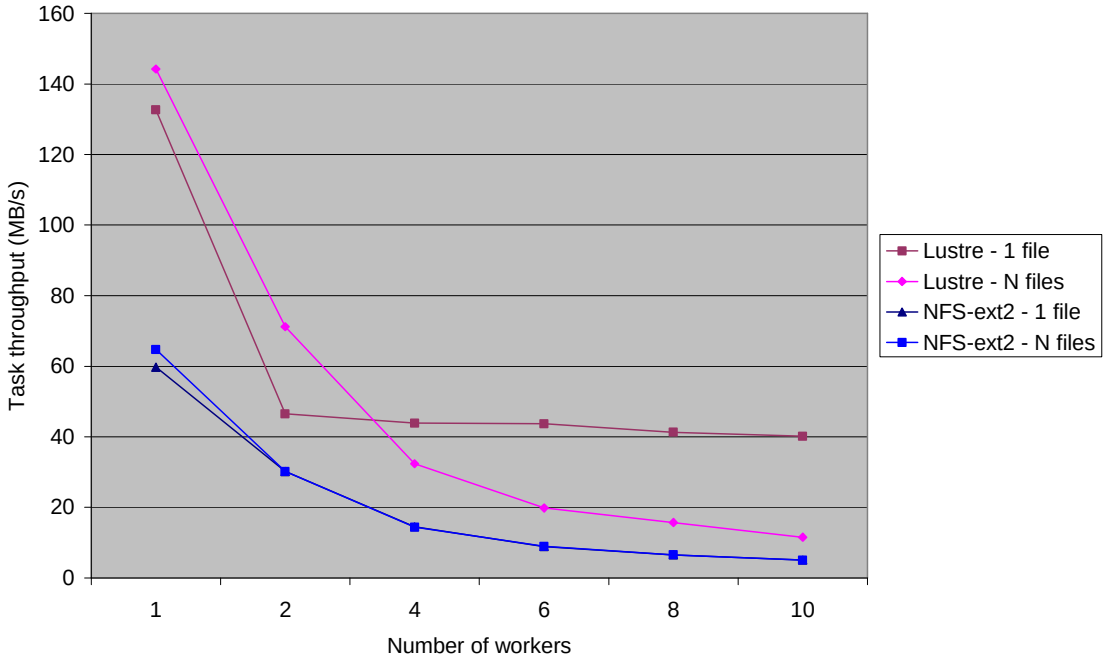

Figure 20 – Throughput per task when various processes read

In Figure 20, we can see that, with Lustre, when working on a single file, from 2 parallel tasks on, each process has a throughput between 40 and 45 MB/s. Even though this throughput tends to slow down because of the client CPU load, it makes the total throughput to be almost proportional to the number of tasks, and it reaches more than 400MB/s with 10 files.

Moreover, the standard deviation is extremely low for this operation: less than 0,002% ! Our hypothesis is that the client reads each part of the file only once, and then serves all the processes waiting for this part. It leads to an amazing gain in efficiency. This is obviously not the case when reading various files: the throughput per task decreases proportionally to the number of tasks, as we have seen for writing.

On the contrary, NFS does not take advantage of the fact that various processes are working on the same file, and we get the same results with a single file as with multiple files.

9.6. Conclusion

Now we know that Lustre is slower than ext3. That was obvious, since Lustre is in fact a layer running over ext3. This makes global performances to be a little bit slower. For small files, the difference is bigger, because of the remote meta-data server, that implies TCP/IP communication overhead.

On the contrary, Lustre has better performances than NFS. So for a remote use, it is highly recommended to use Lustre. We also recommend the use of a network faster than 2 Gbps, since a double Gigabit Ethernet was a bottleneck.

10. Cluster tests

10.1. Computing machines overview

There are basically 3 computing systems to be tested, each one running different operating system and file system. Each user has 2 directories on these computing systems – the home directory and the data directory – this one being provided by another file server through NFS. These directory are accessible from all the machines of the 3 computing systems.

The machines using a SAN are connected through a double 2 Gbps Fiber Channel connection, which implies a theoretical maximum throughput of 4 Gigabits/s.

10.1.1. Calc-gen5

It is composed of a single machine running a Solaris 9 operating system, and the /tmp directory uses the UFS[12] file system, over a SAN. Details are given in Listing 4.

```
Processor  : UltraSparc 4+ - 64 bits (8 dual cores)
Memory     : 32 GB
OS         : Solaris 9
/tmp FS    : UFS – Unix File System
/tmp size  : 132 GB
storage    : SAN
shared ?   : no
```

Listing 4 - Calc-gen5 computing system properties

10.1.2. Calculx

It is composed of 5 machines running AIX 5.3 operating system, named calcul3-ci, calcul4-ci, calcul5-ci, calcul6-ci, and calcul7-ci. The /tmp directory is shared through the General Parallel File System file system (GPFS, [13]). Details are given in Listing 5.

GPFS is a Cluster file system developed by IBM[14]. Its specificity is that it manages by itself the Fiber Channel network, resulting in higher throughput but poorer access times (see 10.3 and 10.5).

Calcul4-ci is furthermore connected by UltraSCSI to a JFS2 directory called /tmp-nastran.

```
Processor  : Power 5/5+ - 64 bits (48 cores in total)
Memory     : 224 GB
OS         : AIX 5.3
/tmp FS    : GPFS          /    JFS2
/tmp size  : 542 GB        /    546 GB
storage    : SAN           /    DAS (UltraSCSI)
shared ?   : yes           /    no
```

Listing 5 - Calculx computing system properties

10.1.3. Linux

This computing system is a Cluster, composed of 48 machines, running Red Hat 4 or 5 operating system. Each machine has its local /tmp directory, with ext3. Moreover, 24 machines have access to a /ptmp directory, shared through the Lustre file system. Details are given in Listing 6.

```
Processor  : AMD - 64 bits (156 cores in total)
Memory     : 608 GB
OS         : Red Hat 4/5
/tmp FS    : ext3          /    Lustre
/tmp size  : 45-203 GB     /    1 098 GB
storage    : DAS           /    SAN
shared ?   : no            /    yes
```

Listing 6 - Linux computing Cluster properties

10.2. Limitations

On each computing system, the maximum authorized file size for the users is 5 GB, so all the tests are driven on files smaller than or equal to 5 GB.

Moreover, on Calculx-ci, the operation ‘lseek’ does not work for files larger than 2 GB. Since ‘lseek’ is required for random reading and writing, these operations will not be tested on this platform for files larger than 2 GB.

It has to be noticed that tests on the computing systems have been run without taking care of other running programs, which could have caused low results.

10.3. Throughput

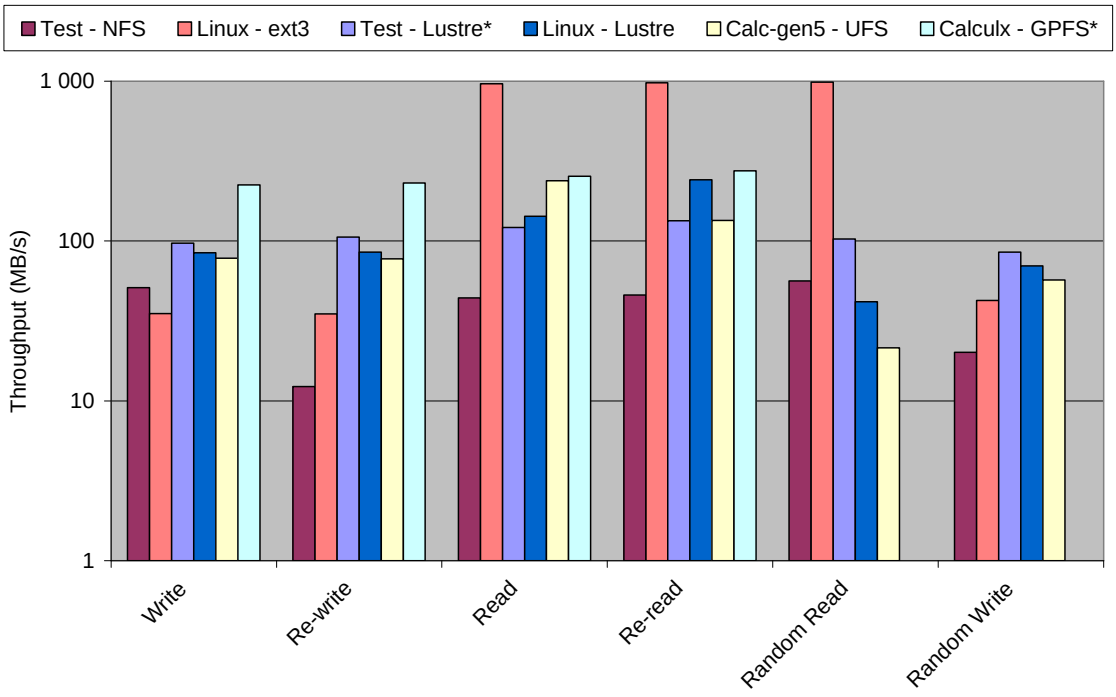

Figure 21 - Computing system throughput tests with a 5 GB file

Note: The results of the test platform are those measured with network bonding (see 9.3.2). The shared file systems are signaled on the graph by a star (*).

As we can see in Figure 21, the objective to avoid any cache effect has been reached in every case but the ext3 on the Linux Cluster. Actually, it is probably related to the fact that it is the only configuration using the internal disk, and not the SAN (JFS2 was not tested there). Moreover, we can notice that, re-writing is as fast as writing, probably because the “pre-empted commit” is disabled on this system.

The system using GPFS performs all sequential operations somewhat faster than all others. For writing, it is at least twice faster than the others. On the contrary, the slowest system for sequential operations is NFS on our test platform.

At last, the slowest file system for writing is NFS. Indeed, the 2 systems (Linux Cluster and the test platform) with NFS are the ones with the slowest performances for this operation. In both cases, using Lustre is around twice faster than using NFS – except in the case of cache effect.

10.4. Cache efficiency

When running the tests on a smaller file (2 GB in this example), we can see on Figure 22 that GPFS seems to be the only system that does not take advantage of the cache. In fact, there is no use of the cache on this system for files bigger than 128 MB.

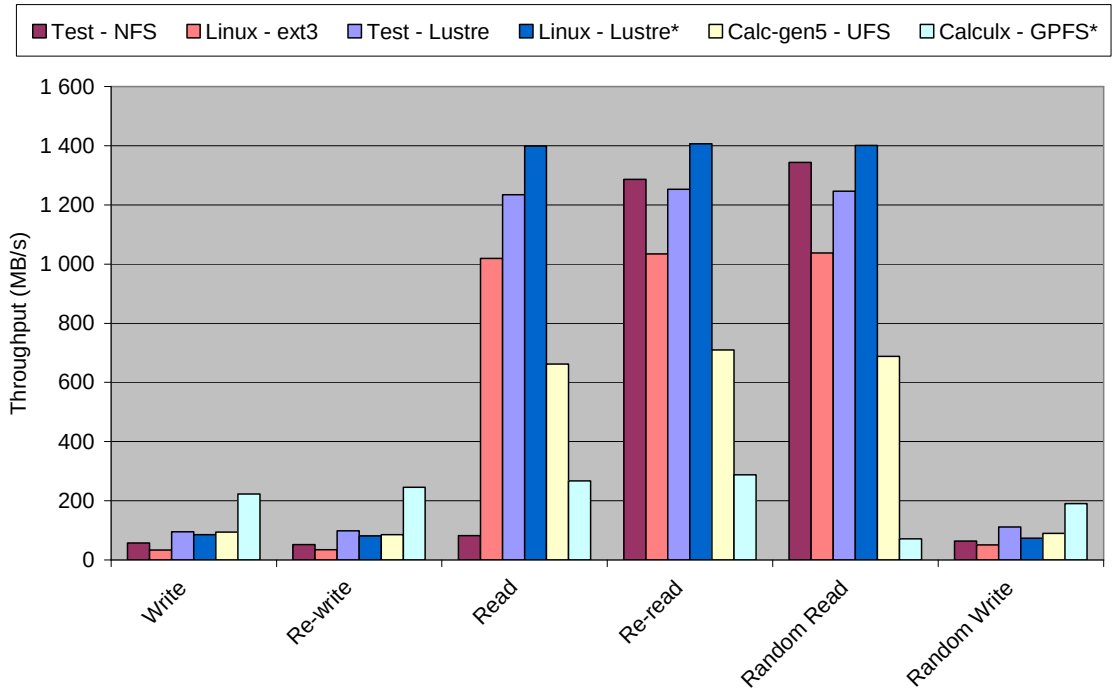

Figure 22 - Computing system throughput tests with a 2 GB file

As we can notice, there is no cache effect for writing, whatever the file system is. Moreover, after writing on NFS, data do not seem to be kept in client memory, since reading then is quite slow, whereas Lustre, ext3 and UFS do it.

Moreover, reading randomly on NFS seems to take a good advantage of the cache, since it is as fast as re-reading, and much faster than the first reading. We may guess that reading randomly

after having cleared the cache would have lead to performances such as the first reading, which is probably not the case with the other file systems.

At last, the faster file system when allowing cache effect seems to be Lustre, that performs almost all the operations faster than UFS, ext3 and NFS.

10.5. Access time

As we can see in Figure 23, two systems have very short access times: UFS/Calc-gen5, and ext3/Linux. However, the result for ext3 is not a surprise: ext3 was yet much faster than Lustre on the test platform.

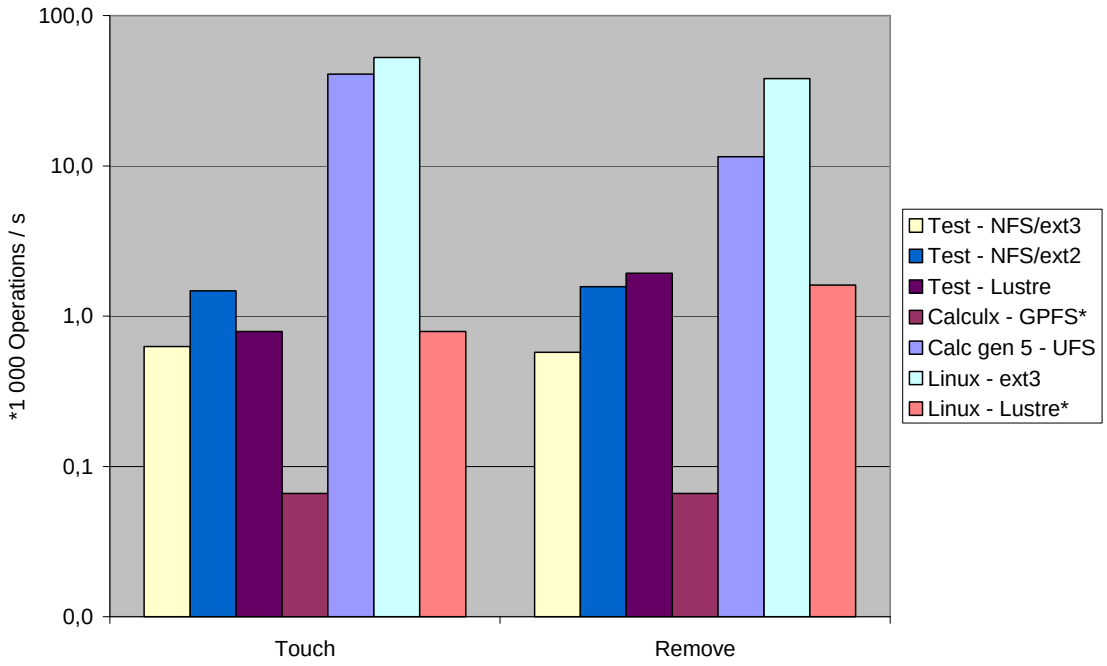

Figure 23 – Computing systems access time efficiency (the higher, the better)

Performances of GPFS/Calculx are very bad: access times for both ‘touch’ and ‘remove’ are around 66 operations per second – 10 times slower than NFS. This is probably due to the structure of the GPFS file system, which needs to make some verifications before writing on the SAN – in particular, to manage concurrent access to the data.

We can observe in Figure 24 that the performances got from the Linux Cluster are surprisingly similar to the ones from the test platform.

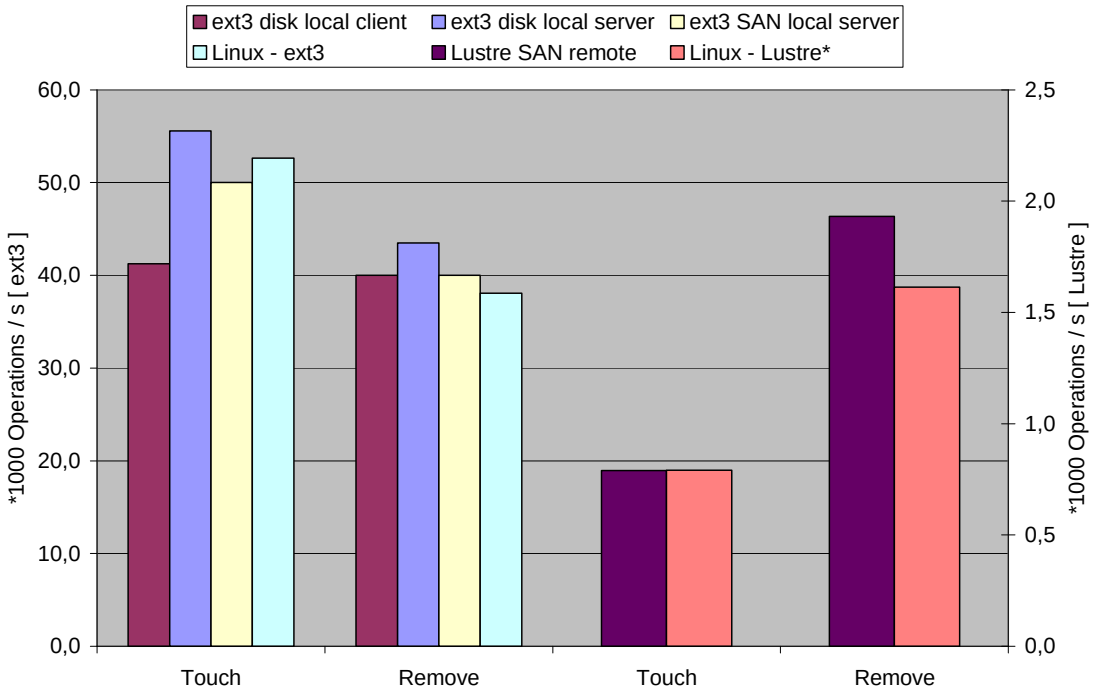

Figure 24 - Test platform and Linux Cluster comparison (the higher, the better)

10.6. Conclusion

Since these tests have been driven on different configurations, we can only compare the combinations hardware + file system.

It looks like GPFS/Calculx is a good solution for a system with big files, on which we will not use the cache. Indeed, it has very high access times (bad to work with small files), a very good throughput (good for big files) and it never uses the cache.

For a system which will take a good advantage of the cache, UFS/Calc-gen5, or Lustre/Linux, are better solutions. The first one is typically made for small files, because it has very low access times, whereas Lustre/Linux, which has better cache and throughput performances, is designed for bigger files.

Ext3/Linux could be a good compromise for systems with big and small files, and provides a very efficient access to the cache. However, it is a local file system, and using it remotely severely decreases its efficiency.

At last, we saw that our test platform performances were very similar to Linux Cluster performances. Thus we can consider extending all the results obtained in paragraph 8 to the Linux Cluster.

11. Configuration tests

11.1. Scheduler

11.1.1. Test overview

In this part we will test I/O scheduler, as suggested in the Lustre Operations Manual [10]. The four schedulers that are included in the version 2.6 of the Linux kernel, shall be tested: Completely Fair Scheduling (CFQ), Deadline, Anticipatory Scheduling[15], and Noop. The Lustre Operations Manual recommends using the Deadline schedulers, to reach the best efficiency.

I/O schedulers are not invoked when accessing data in the cache, so we will not make cache efficiency tests. Moreover, to simplify the process, throughput and access time tests will be driven with single clients.

11.1.2. Client scheduler

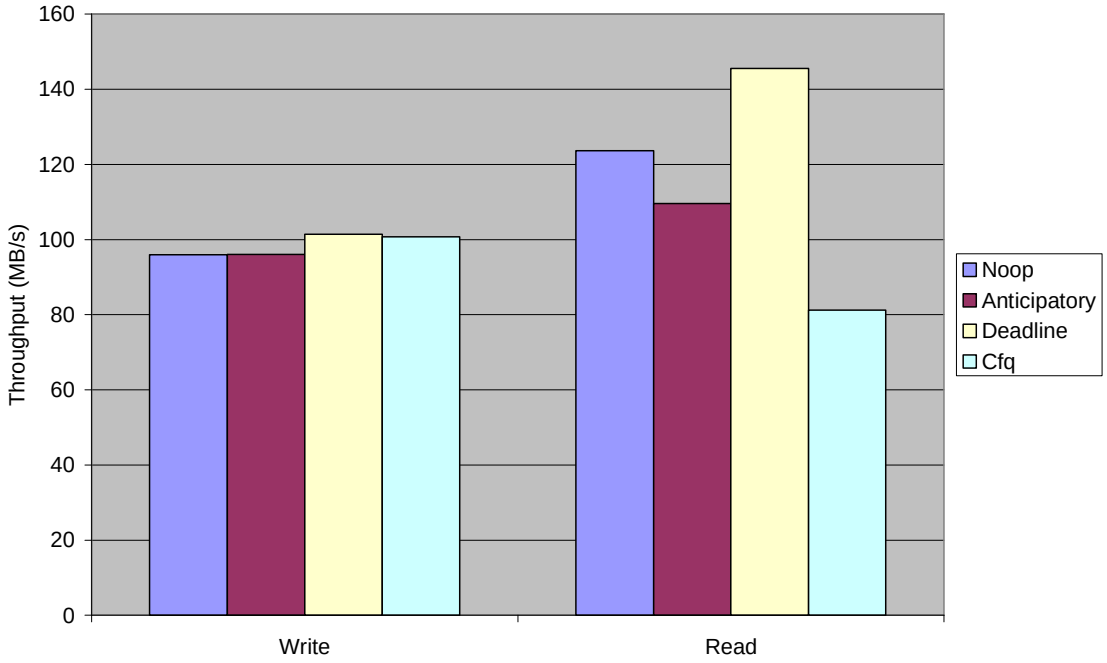

Figure 25 – Throughput with the different client I/O schedulers

As explained in the documentation, CFQ gives the lowest results, what we can see on Figure 25. This is hard to see for writings, but it becomes obvious with readings. Noop and Anticipatory seem to be a little bit faster, and the best is obviously the Deadline scheduler... as expected.

For the access time, differences of performance are to small (< 7%) to be significant, even though Anticipatory is a little bit slower than the other ones.

11.1.3. OST server scheduler

Running the tests locally on the OST server, the 4 different schedulers had very similar performances. The mean difference of performances always stayed below 5%, except in one case: Deadline scheduler seemed to remove files around 15% faster than the others.

With remote tests, the Deadline scheduler seems to be, once again, the best choice, since it performs both reading and writing faster than the others ones.

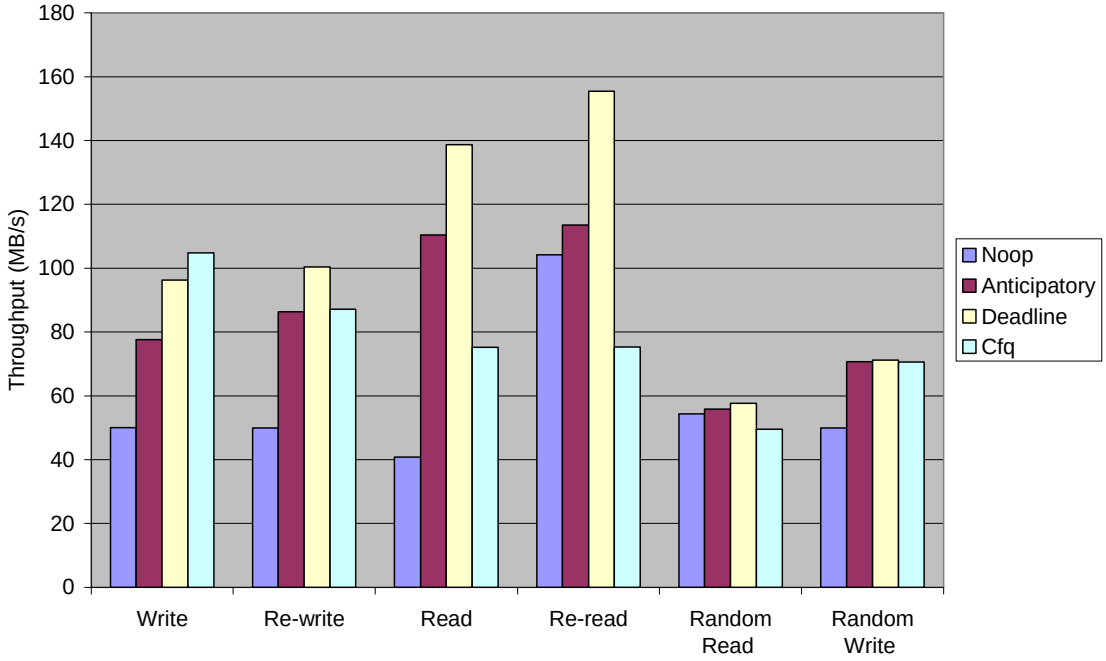

Figure 26 – Remote throughput with the different OST server I/O schedulers

It can be noticed that CFQ performs readings slower than writings, and it is the only scheduler that does not take advantage of the cache for re-doing an operation. At last, we can see that random operations are performe almost as fast by the 4 schedulers – CFQ is a little bit slower for reading, and Noop is quite slower for writing.

Concerning the access times, differences of performance are lower than 5%, so the scheduler seems to have a very low impact on small files performances.

11.2. Client I/O RPC Stream tunables

The maximum number of pages that will undergo I/O in a single RPC to the OST could not be greater than 256 on our test platform. The value can be modified by changing the following file: **`/proc/fs/lustre/osc/<object name>/max_pages_per_rpc`**.
Whatever the value of this variable is, the access times remain unchanged, but the throughput increases linearly with the number of pages, until we reach the maximum.

The RPCs in flight number corresponds to the maximum number of tasks that can be managed concurrently by the server. The value is in the following file:
`/proc/fs/lustre/osc/<object name>/max_rpcs_in_flight`.
This number must be big enough to avoid starvation. With very low numbers (e.g. 1 RPC in flight), the throughput with a single task was quite low (85MB/s for writing, 60MB/s for reading). On the contrary, high number may lead to overload, what we could not have tested because we did not have enough clients to saturate the server.

11.3. Different Lustre configurations

We tested different configurations of Lustre during the internship. We will explain them here, and see what are their advantages and disadvantages.

The first configuration tested is the classic one, composed of one MGS/MDT server, and one OST server. It seems to work well, and we recommend this configuration for the later use of Lustre in computing Clusters. Since we guess there will not be a lot of Lustre systems, installing the MGS apart seems to be irrelevant.

The other configuration tested is a configuration with all the services on the same machine: MGS, MDT and OST. Notice that Lustre is designed to prevent anyone of using this configuration. Indeed, even though it may have some advantages, mainly for the access times, this configuration may lead to some problems, because the same machine will have to handle with twice more operations: data and metadata storage.

12. Synthesis

As we have seen all along the tests performed, Lustre is not recommended for a local use, since ext2 and ext3 have much better results. On the contrary, Lustre has better performances than NFS, and it offers a scalability that NFS cannot give. Hence, Lustre seems to be a very good solution for a file system intending to allow data sharing, in a very big quantity.

Moreover, we only tested during this internship platforms where clients and servers are connected through an Ethernet connection. As we have seen, Lustre is limited by this Ethernet connection, even when we made bonding. Since Lustre can manage Inifiniband networks, that are much faster than Ethernet ones, we can assume that performing the tests over Infiniband will improve the performances.

Infiniband is a technology that is already used at the CNES, but we had not the time to perform the tests again with this technology. We strongly recommend to make these tests before installing Lustre in any computing system, since they can show very interesting improvements..

13. Time repartition

We can see in Figure 27 the original planning, and in Figure 28 the time spent on each task.

The main reason why the time table has been modified is the early arrival of the SAN. So we did not perform disk tests, and then SAN tests, but we perform them in parallel. Moreover, we did not perform GPFS tests on the test platform, because of the complications we had to install it, and we changed the order of configuration tests and Computing system tests. At last, we performed ext2 and FTP tests quite late, since they were not included in the original test protocol, and we performed other parallel tests at the end of the internship because the results we had from the first tests were not relevant at all.

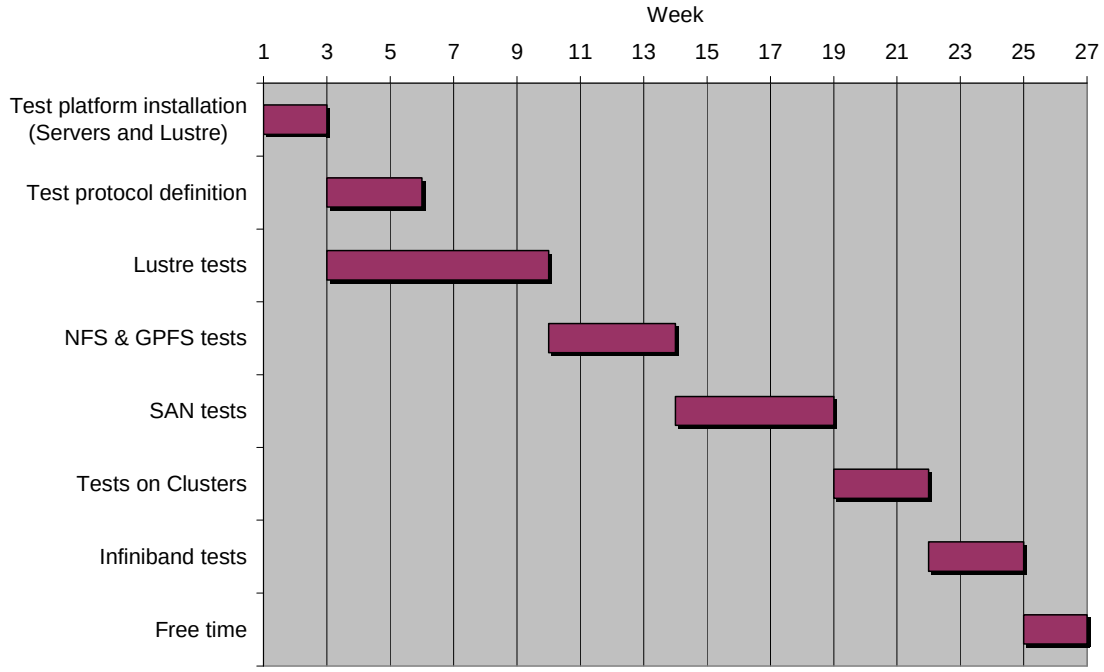

Figure 27 - Time table planning

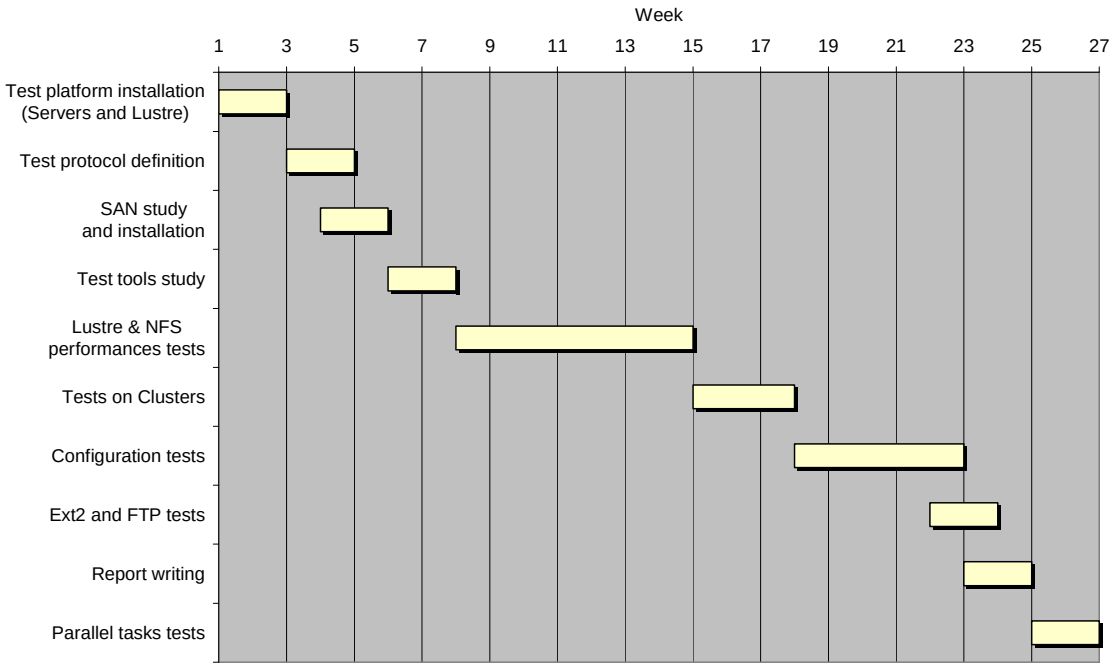

Figure 28 - Effective time repartition

14. Conclusion

This internship was very rewarding, for different reasons.

The first reason is obviously the technical aspect. Indeed, I could acquire new technical competences, on the following systems:

- first of all is Lustre, of course, and its way of working: separation of data and metadata, failover mechanism, communication between the different nodes, etc.
- then comes the SAN (Storage Area Network). I had never heard about this technology before, and my idea is that it is essential to know about SAN, NAS and DAS, for someone willing to work in the domain of the operating systems.
- I learned also the basis about computing computing systems, and how to use it.
- I also discovered the Red Hat operating system, along with the GPFS and UFS file systems, on which I had never worked before.
- at last, I acquaint myself with some quite simple measurement tools, the most important being the Iozone file system benchmark, and the 'dd' Unix program.

The second point is, I could observe some of the management troubles an enterprise can live through, and understand how vital it is to have a well organized management team.

Last but not least, it was a good experience to me to work independently. It was the first time I practised self-working for a quite long period, and I had to define the test methodology, make my own planning and organize myself. This is in my mind something very important, since an engineer must be prepared to self-working, but also, to manage the tasks and time repartitions within team members, when he is lead to be a project manager.

About the firm, I can say the CNES is a quite good firm to work in, with very interesting, innovative and motivating project, especially for someone who is very attracted by aeronautics and space like I am.

15. References

[1] Red Hat. "Red Hat Enterprise Linux 5". Red Hat Company Website. http://www.redhat.com/rhel/. Retrieved March, 06. 2008.

[2] Free Software Foundation. "GNU General Public License". Free Software Foundation Website. http://www.fsf.org/licensing/licenses/gpl.html. Modified February, 12. 2008. Retrieved February, 23. 2008.

[3] "Sun Microsystems". Sun Microsystems Website. http://www.sun.com. Retrieved February, 25. 2008.

[4] "Sun Welcomes Cluster File Systems Customers And Partners". Sun Microsystems Website. http://www.sun.com/software/Clusterfs/index.xml. Retrieved March, 13. 2008.

[5] "Lustre File System – Overview", Sun Microsystems Website. http://www.sun.com/software/products/lustre/index.xml. Retrieved July, 11. 2008.

[6] "Lustre File System – Get the Software", Sun Microsystems Website. http://www.sun.com/software/products/lustre/get.jsp. Retrieved April, 11. 2008.

[7] Philippe Daix (APX Synstar), "Dossier d'installation", for the CNES, November, 28. 2007, Chapters 7 & 8

[8] Qlogic. "Qlogic Drivers". Qlogic Company Website. http://support.qlogic.com/support/drivers_software.aspx. Retrieved March, 20. 2008.

[9] Lustre 1.6 Operations Manual, version 1.10, December 2007, Chapter 21

[10] Lustre 1.6 Operations Manual, version 1.10, December 2007, Chapter 3.3.4

[11] Lustre Features Roadmap, version 1.20, December 2007

[12] Sun Microsystems. “Unix File System (UFS) at OpenSolaris.org”. OpenSolaris Website. http://opensolaris.org/os/community/ufs/. Retrieved May, 26. 2008.

[13] Frank Schmuck and Roger Haskin, “GPFS: A Shared-Disk File System for Large Computing Clusters”, Proceedings of the FAST 2002 Conference on File and Storage Technologies, January, 28-30. 2002.

[14] “IBM General Parallel File System”, IBM Website. http://www.ibm.com/systems/clusters/software/gpfs/. Retrieved July, 29. 2008.

[15] Sitaram Iyer and Peter Druschel, “Anticipatory scheduling: A disk scheduling framework to overcome deceptive idleness in synchronous I/O”, ACM SOSP 2001, August, 6. 2001

[16] Lustre 1.6 Operations Manual, version 1.10, December 2007, Chapter 1.2.6, Figure 1.1

[17] “Change Log 1.6 – Wiki3”, Sun Microsystems – Lustre Wiki Website. http://wiki.lustre.org/index.php?title=Change_Log_1.6. Retrieved August, 18. 2008.

APPENDIX

APPENDIX I
PERFORMANCE MEASUREMENT TOOLS

In this appendix we will explain the different tools and commands used to measure the performances of the file systems.

1. Iozone[6]

This benchmark tool covers a grand variety of file systems operations. Moreover, it provides some interesting options, among which the possibility to perform tests from different clients simultaneously.

We used the version 3.287 of the benchmark.

1.1. Main options

The Iozone settings and each operation test are explained in the Iozone documentation[7]. The operations that can be tested with Iozone are given with their reference in the Listing 7. The automated test mode includes operations 0 to 7.

```
0=write/rewrite, 1=read/re-read, 2=random-read/write, 3=Read-backwards, 4=Re-write-record, 5=stride-read, 6=fwrite/re-fwrite, 7=fread/Re-fread, 8=random mix, 9=pwrite/Re-pwrite, 10=pread/Re-pread, 11=pwritev/Re-pwritev, 12=preadv/Re-preadv.
```

Listing 7 - Iozone operations number

Let us explain the main options to run an Iozone test (all the options are explained in the Iozone documentation).

The '**-c**' option allows including the **close()** operation in the timing calculations. As said in the documentation, this can be useful for NFS.
The '**-e**' option includes flush in the timing calculations.

The '**-U /mnt/fs/**' option would have been very useful since it would have forced the system to unmount and remount the mountpoint between each test, and thus guarantee that the buffer cache does not contain any of the file under test. Unfortunately, this option never worked on our platform.

In our platform, none of the atomic operations (**pread**, **pwrite**, **preadv** and **pwritev**) worked. Atomic operation tests are available for just a few platforms (). Since **pread** seemed to be installed and working fine, we tried to force the compilation with the **HAVE_PREAD** flag. It still didn't work, and wasn't that important (others read and write are tested), so we let it pass.

[6] Information about the Iozone file system benchmark can be found on the Iozone website: "Iozone Filesystem Benchmark". Iozone project webpage. http://www.iozone.org. Modified October, 28. 2006. Retrieved March, 14. 2008..
[7] William D. Norcott, "Iozone Filesystem Benchmark". Available at http://www.iozone.org/docs/Iozone_msword_98.pdf. Retrieved April, 4. 2008

As the full automatic mode ('**-a**' option) performs all the others tests except the random mix, we decided to run the tests with this option, when possible, and to perform the random mix test later.

To run the automatic tests on file sizes up to 4 GB, and with various block sizes, we ran the command given in Listing 8.

```
iozone -g 4G -f /mnt/fs/test_tmp -Rcezab output.wks > output.txt
2> errors.txt
```

Listing 8 - Iozone command example for automatic tests

To perform a test with the basic operations on a specific file size, and with a specific block size, we ran the command given in Listing 9.

```
iozone -r 1024k -s 8g -f /mnt/fs/test8G_tmp -Raceb output.wks
> output.txt 2> errors.txt
```

Listing 9 - Iozone command example for a specific file size

1.2. Multi-process and multi-threads

The '**-a**' option cannot be used to perform tests with several process or threads. It must then be replaced by a list of '**-i**' option, each one specifying an operation to test (basically all operations from 0 to 7).

To run several threads on the same machine, the '**-t**' option is used, with the number of threads required, and the '**–f**' option must be replaced by '**–F**', followed by one filename per thread.

To run the tests from different machines, use the option '**-+m**', as explained in the documentation. Make sure **rsh** is working between the machines. Here the random mix (test number 8) can be tested, and we can fix the percentage of reading through the '**-+p**' option.

At last, the '**-+t**' option, used together with the '**-+m**' one, was supposed to perform network performance tests. This would have been very useful for the tests on remote clients, to evaluate the impact of the network. Unfortunately, it didn't work.

2. The 'dd' command

The '**dd**' command is a program under the GNU licence, provided by Unix whose purpose is low-level copying of data. We used the version 5.97.

The main options are:

- '**if**' to specify the input file. We basically used the **/dev/zero** pseudo-file as input when performing writes to the disk.
- '**of**' to specify the output file. We basically used the **/dev/null** pseudo-file as output when performing reads from the disk.
- '**bs**' is the size of each chunk to be read and written.
- '**count**' is the number of chunks we want to read and write.

Both choices of **/dev/zero** and **/dev/null** as input and output files made the result to depend only on one disk operation, since calls to **/dev/zero** and **/dev/null** do not require any access to the disk, and are very fast, as we can see in Listing 10.

```
# dd of=/dev/null if=/dev/zero bs=1024k count=8192
8192+0 enregistrements lus
8192+0 enregistrements écrits
8589934592 octets (8,6 GB) copiés, 0,279429 seconde, 30,7 GB/s
```

Listing 10 - 'dd' command efficiency

To perform 8 GB reads/writes by 1 MB chunk, the commands used are given in Listing 11.

```
dd if=/dev/zero of=/mnt/fs/output.tmp bs=1024k count=8192
dd if=/mnt/fs/input.tmp  of=/dev/null bs=1024k count=8192
```

Listing 11 - 'dd' commands to read and write 8 GB files

3. FTP throughput

To measure the throughput with the FTP protocol, and in order to depend on only one device, we used a combination of both '**ftp**' and '**dd**' Unix commands.

The command used is a recommendation of IBM for AIX systems to analyse network performances[8], and it seems to work on Red Hat platforms as well. The idea is to take the result of the '**dd**' command as input, to write it on the desired file system. This can be achieved through the command given in Listing 12.

```
ftp> put "|dd if=/dev/zero bs=1024k count=8192" /mnt/fs/test.tmp
```

Listing 12 - FTP command to write a file

To test the network performances only, without taking care of the file system, just write in the **/dev/null** pseudo-file instead of a real file. The command used is given in Listing 13.

```
ftp> put "|dd if=/dev/zero bs=1024k count=8192" /dev/null
```

Listing 13 - FTP command to test the network throughput

At last, to perform reads, we just have to use the '**get**' FTP command, and write the result in **/dev/null**, as explained in Listing 14.

```
ftp> get /mnt/fs/test.tmp /dev/null
```

Listing 14 - FTP command for reading

[8] The source can be found at the following address, which has been retrieved on july, 16. 2008: http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/com.ibm.aix.prftungd/doc/prftungd/ftp.htm

4. Access time measuring commands

To measure the mean access time on each file system, we used the commands '**touch**' – to access empty files – and '**rm**'– to delete them.

Since the creation and deletion of a single file is too fast to put in evidence differences between the file systems, we created and removed 1 000 files in each test. But the '**touch**' command does not allow us to create many files in one system call, so we will first create the files through a loop, as we can see in Listing 15.

```
time for ((i=0; i<=1000; i=i+1)) ; do
     touch /mnt/fs/test$i.tmp
done
```

Listing 15 - Loop to create the test files

Therefore, we will be able to touch these files or remove them in one command, as explained in Listing 16.

```
time touch /mnt/fs/test*.tmp
time rm -f /mnt/fs/test*.tmp
```

Listing 16 - 'touch' and 'remove' commands

5. I/O Kit

Lustre is provided with the I/O Kit[9], that allows the user to have more information about the file system behaviour. We installed it on one machine to have an idea of how it worked. However, we did not use it for our experiments since its mechanism is a little bit complicated, and we could get the data we wanted in a quite simple way.

[9] Lustre 1.6 Operations Manual, version 1.10, December 2007, Chapter 19

APPENDIX II
COMPARING IOZONE AND 'DD' RESULTS

For some tests, we got two sets of measures, one from Iozone, and the other one from 'dd'. The idea was to compare the results, to avoid measurements errors.

In this section, we will compare these different results we had for some operations – mainly throughput tests, but also cache efficiency tests. Since 'dd' only performs sequential reads and writes, we will not talk about random operations.

1. Local throughput

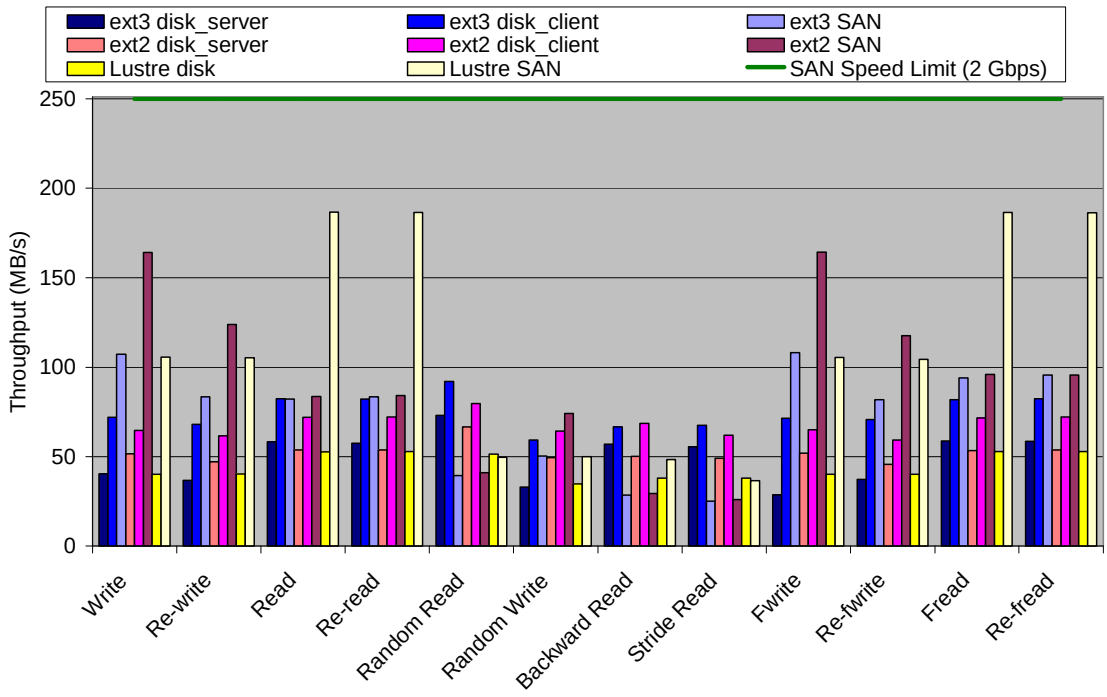

Figure 29 – local throughput test, measured with Iozone

The strange thing in this result is that re-writing on ext3 seems to be slower than writing, on the Iozone results – from 5 to 9% when writing on the internal disk, up to 22% on the SAN. This is caused by the ext3 pre-empted commit, because re-writes were performed just after writes, without any latency. When the second write starts, it has to wait for the data of the first write to be written, thus leading to a delay.

With these Iozone results, we could make some observations, which will be confirmed further by the 'dd' results:

- The client platform is faster than the server. Server performances are always around 25% under client ones.
- Writing on the SAN is at least twice more efficient than writing on the internal disk, whatever the file system.

- When reading, using Lustre on a SAN seems to be really more efficient than any other configuration.
- On the ext3 file system, writing is faster than reading – except for random read/write on the internal disk.

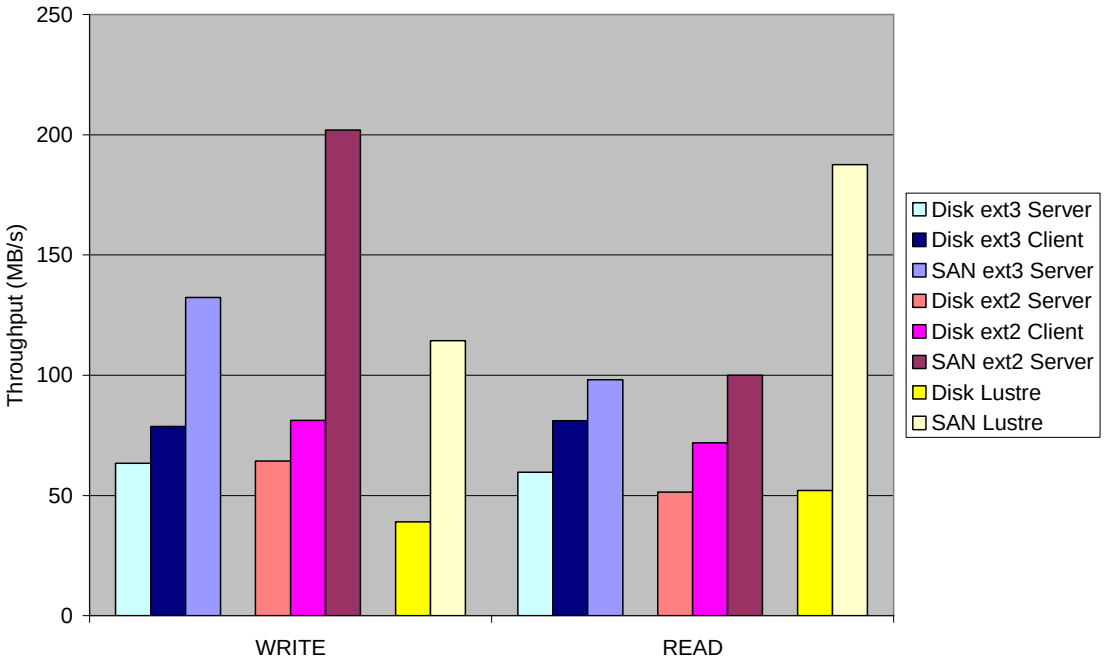

Figure 30 – Local throughput test, measured with 'dd'

The only contradiction between both experiences is that writing on an ext3 file system is not always faster than writing on Lustre. When using the SAN, the operations of re-writing are even quite worse with ext3 than with Lustre. This, and the fact that reading was slower than writing, was very astonishing. Thus we had to check where the problem could come from, and find out the ext3 “pre-empted commit” effect.

At last we can see that ‘dd’ performances are always a little bit faster than Iozone ones, probably because of the running application overhead.

2. Remote throughput

On these two graphs, performances are very similar. The only difference is that reading on NFS over the SAN gives better results with the ‘dd’ command – around 75 MB/s – than with the Iozone tool – 45 MB/s only, not faster than writing.

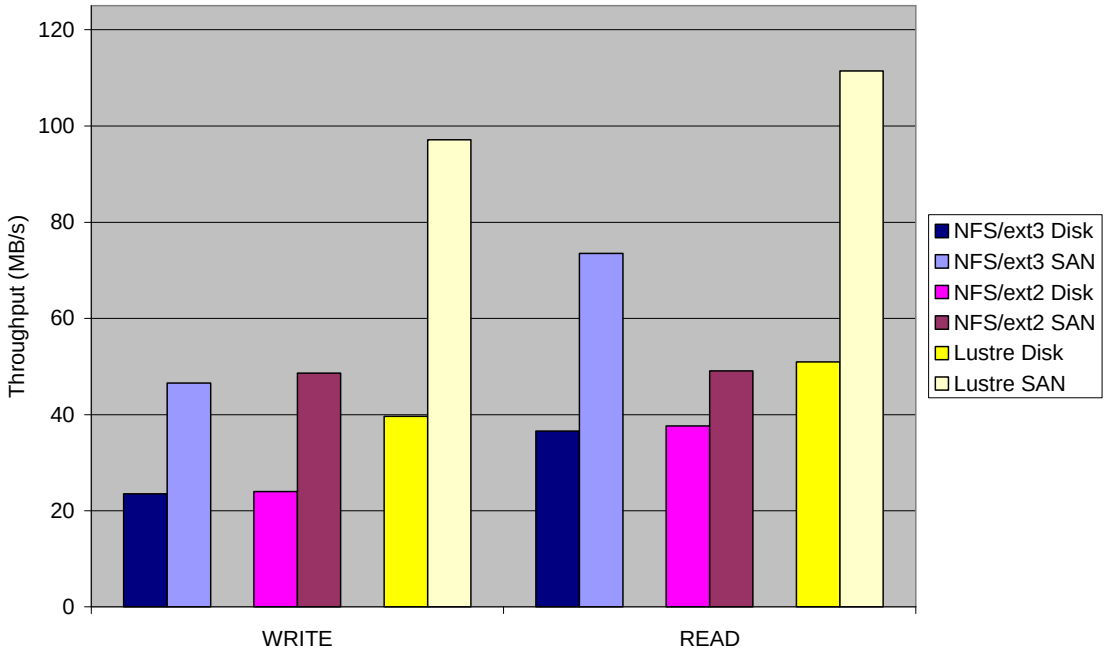

Figure 31 - Remote throughput test, measured with 'dd'

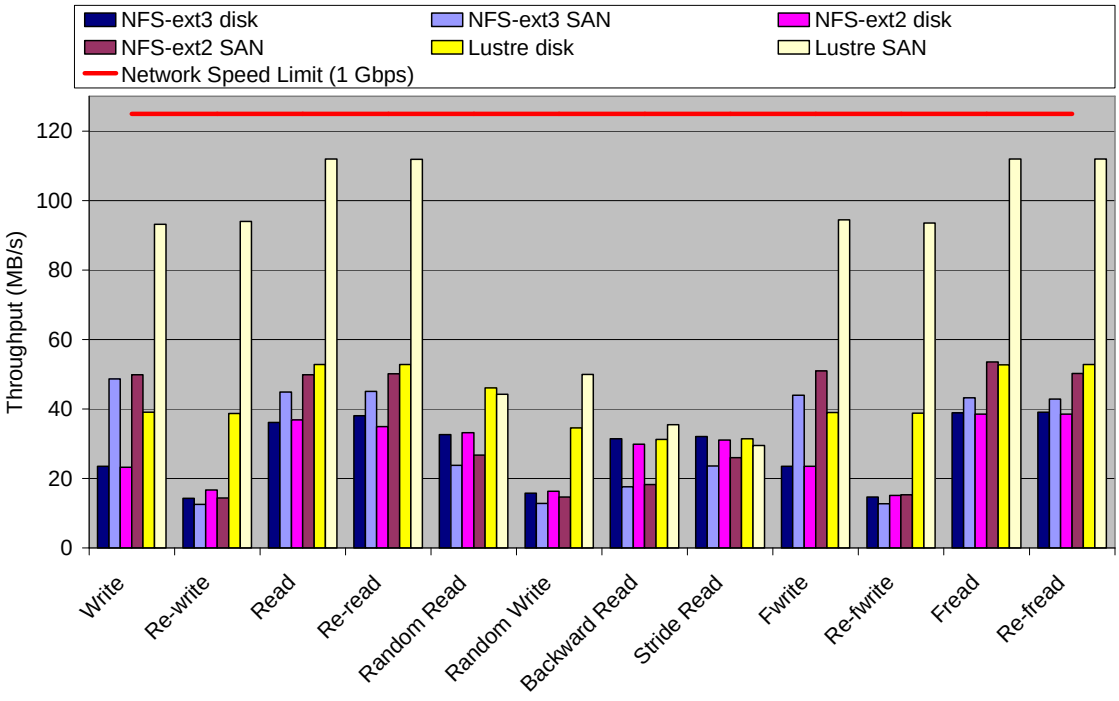

Figure 32 - Remote throughput test, measured with Iozone

3. Schedulers

3.1. Client scheduler

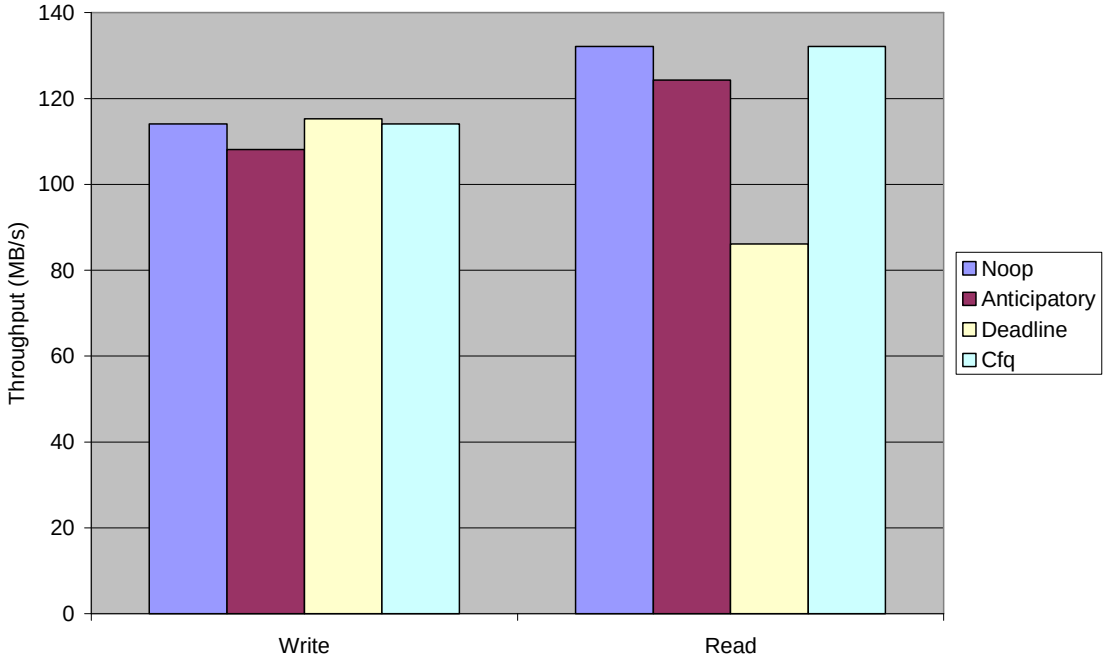

Figure 33 - Client scheduler test: 'dd' results

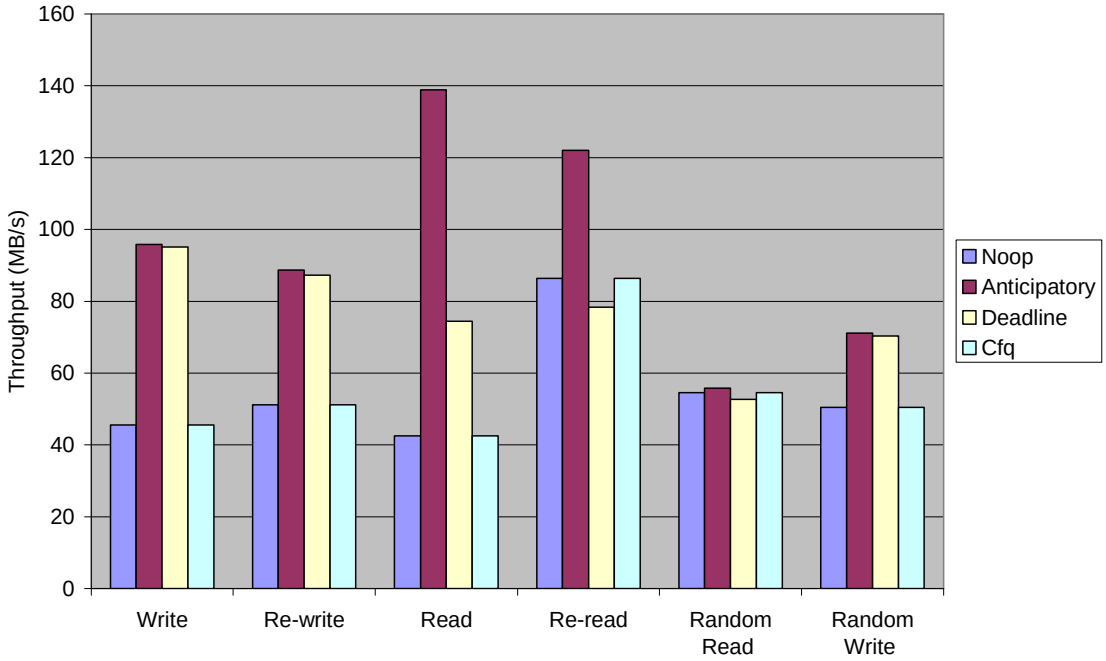

Figure 34 - Client scheduler test: Iozone results

For this tests, ‘dd’ and Iozone results are incredibly different. In the first one, Noop performs quite well, Anticipatory is from far the slowest, Deadline is the best, and CFQ writes faster than it reads.

The Iozone results says the contrary: Noop is from far the slowest, Anticipatory is the best one, Deadline writes faster than it reads, and CFQ is almost as fast as Anticipatory.
This is totally incredible.

3.2. OST server scheduler

On the contrary, when testing the different OST server schedulers remotely, ‘dd’ and Iozone results were quite coherent, even though they differ a little bit.

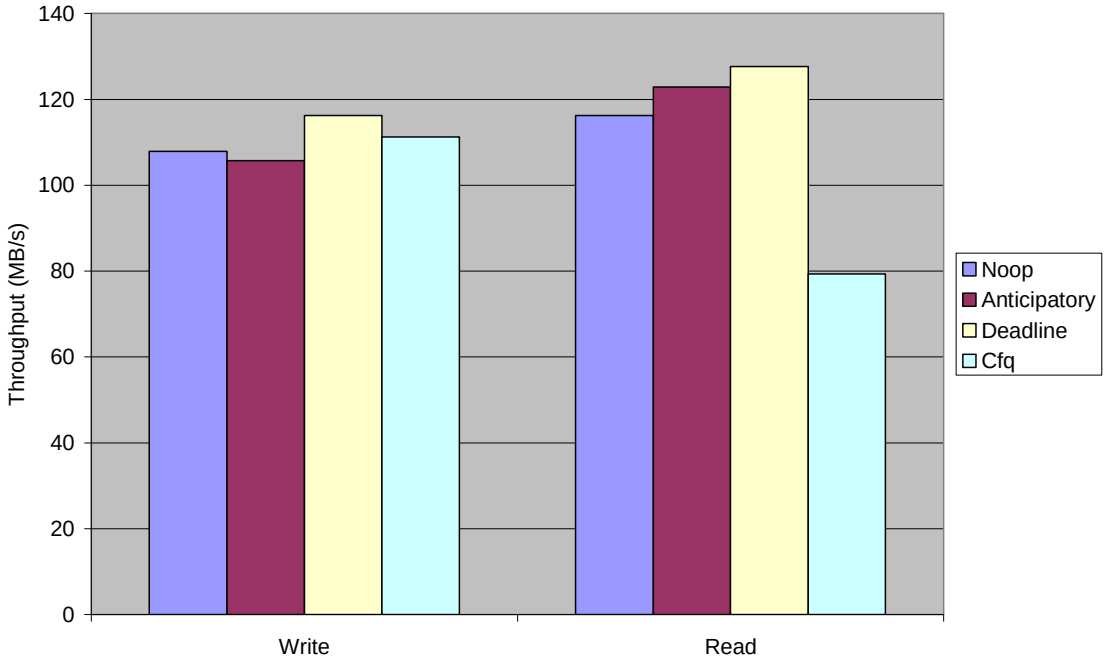

Figure 35 - OST server scheduler remote test: 'dd' results

The coherence lies in the global shape of the results. In both case, Deadline is the best scheduler, before Anticipatory and Noop, and in both case CFQ writes quite quickly, but reads slower.

However, there is some difference in terms of values. When writing, the ‘dd’ results are much higher than the Iozone ones – up to 2,5 times for the Noop scheduler. It is the same case again for reading with the Noop and Anticipatory schedulers.

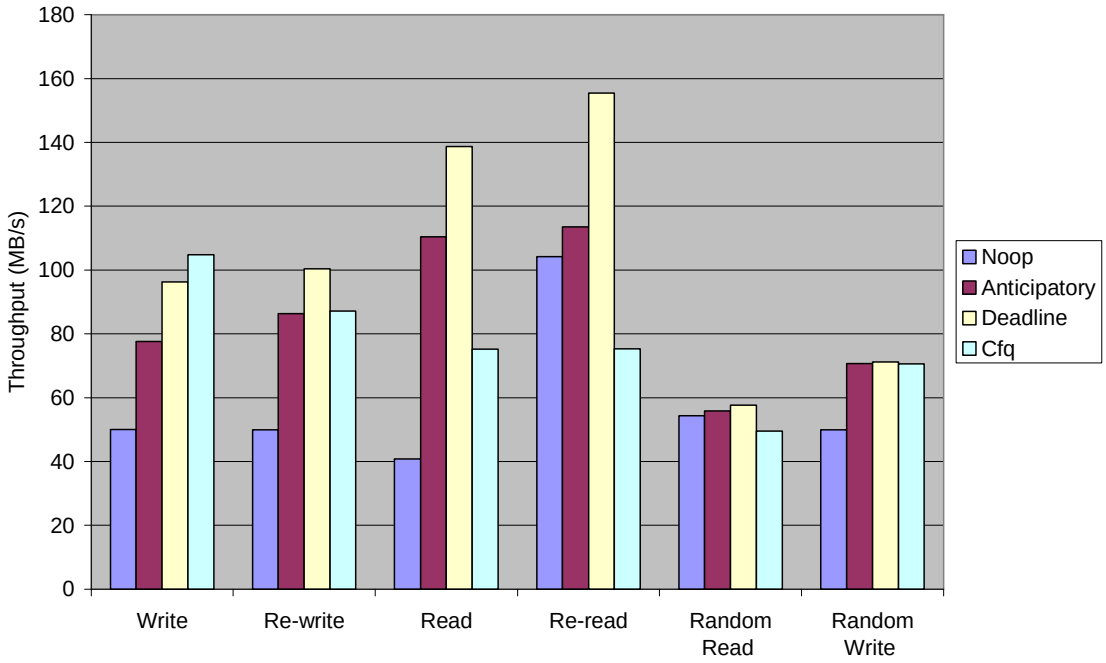

Figure 36 - OST server scheduler remote test: Iozone results

APPENDIX III
MORE RESULTS…

In this appendix we present different results that we have observed, and that do not concern directly the aim of the internship.

These results are basically:

- Internal disk and SAN performances comparison
- Client and server machines comparison – that is, comparison between IBM xSeries 336 and SunFire x4200

1. Comparing disk and SAN results

1.1. Buffer cache

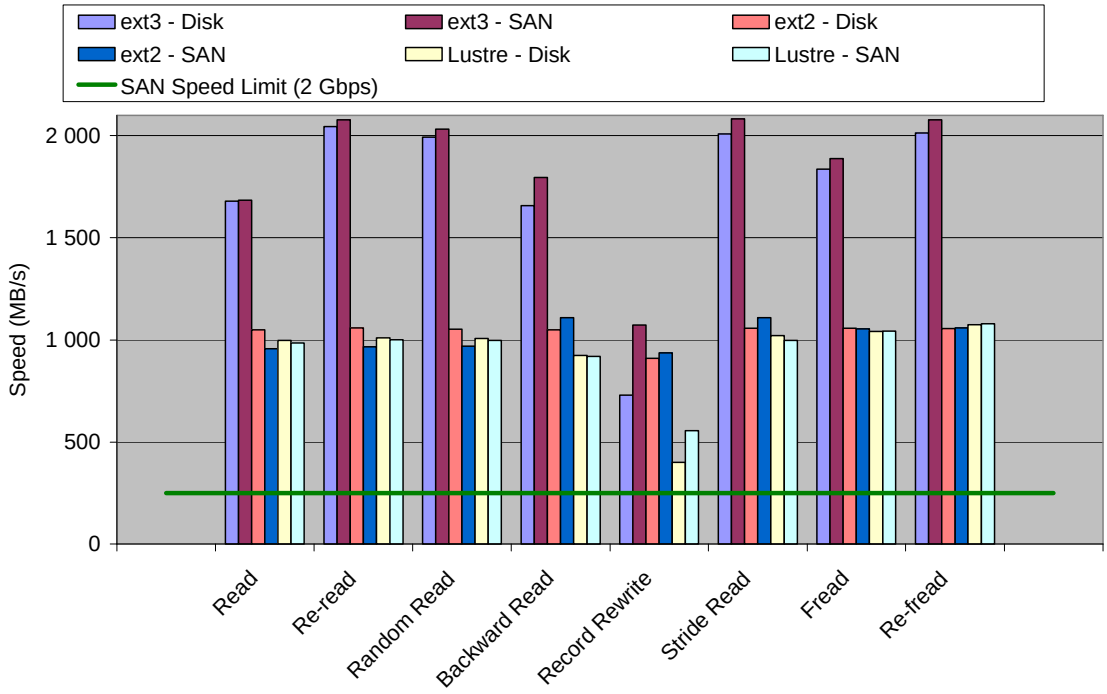

Figure 37 - Local cache tests on the server

On the two graphs, we just printed out the results in which the RAM is involved. However, some operations for which we expected there would be some cache interference did not take advantage of it, as we can see on the remote graph for the NFS/ext2 file system.

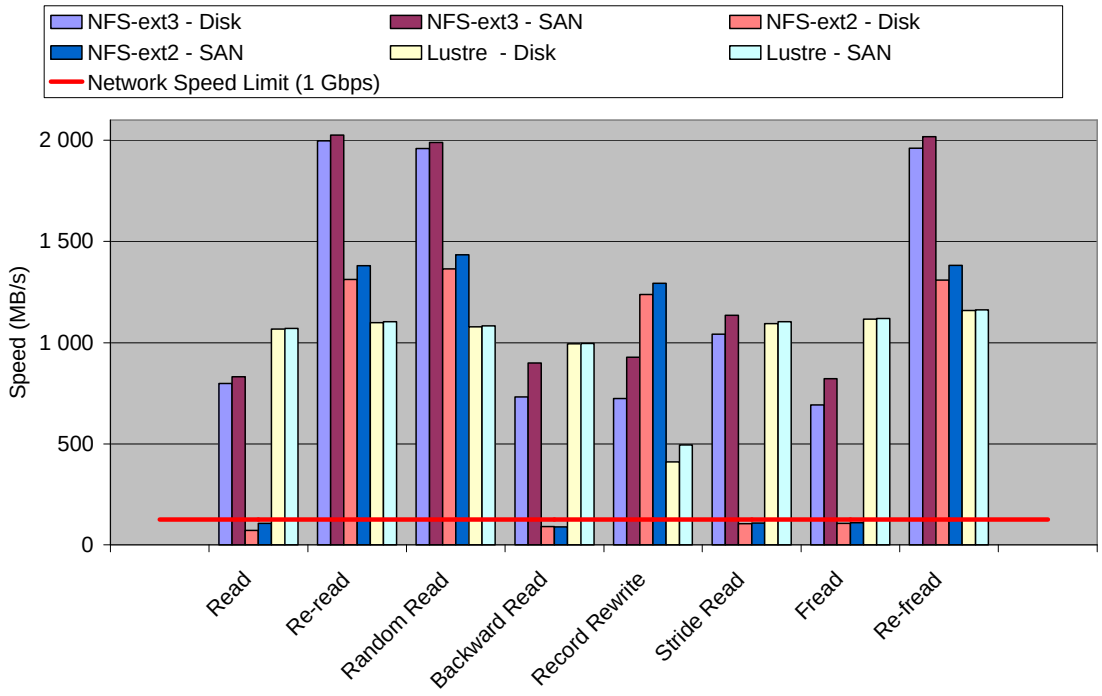

Figure 38 - Remote cache tests

As we can see in the two previous graphics, using the internal disk or the SAN has almost no impact on cache results. This is obviously what we expected, since the storage device is supposed to little interfere in the process.

1.2. Local throughput

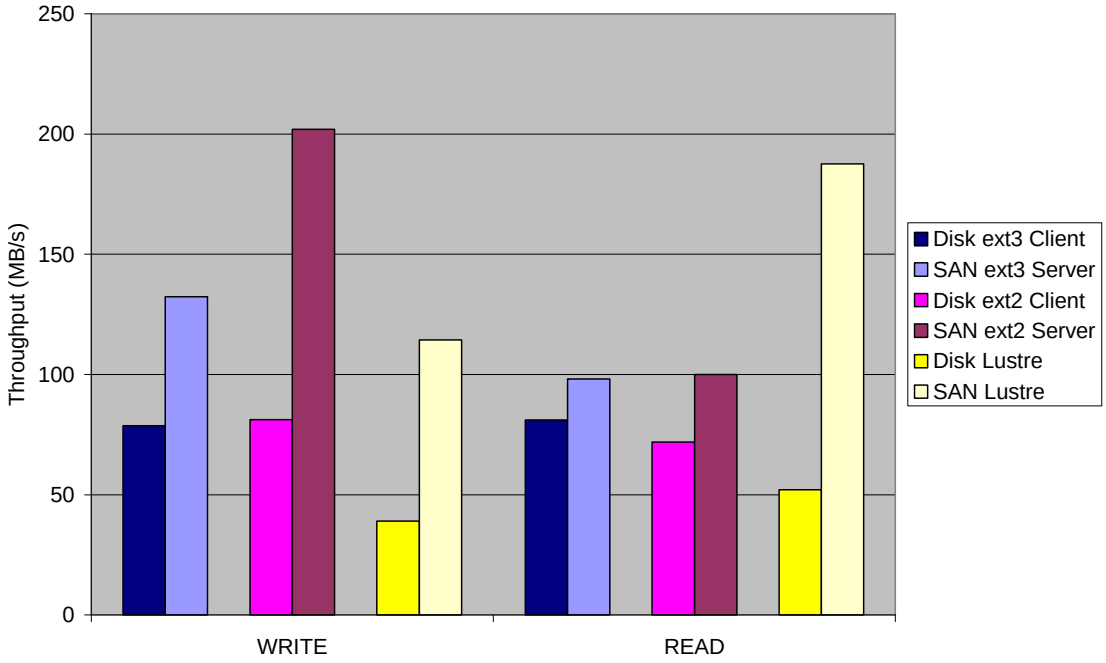

Figure 39 - Local throughput

First of all, we can notice that writing on a SAN is significantly faster than writing on the internal disk, as we expected. Indeed, ext3 writes twice faster on the SAN than on the disk, and it is even three times faster in the case of Lustre and ext2.

Even though we could think writing on ext2 is limited by the Fiber Channel network of the SAN, it is not the case, since the real throughput on this network is around 175 MB/s. If this network was saturated, we would observe a throughput higher than 250 MB/s, due to the 'pre-empted commit' effect.

When reading, the contrast is slightest for ext2 and ext3: the SAN is less than twice faster. On the contrary, Lustre performs read almost 4 times faster, and is probably limited by the Fiber Channel network.

Then we can guess that the maximum throughput to the internal disk of the server is lower than 60 MB/s, since no file system wrote on or read from it faster (even though both ext2 and ext3 seems to write faster). In like manner, we can guess that the maximum throughput of the client disk is lower than 80 MB/s.

At last we can observe that both ext2 and ext3 are better than Lustre when we used the internal disk.

1.3. Remote throughput

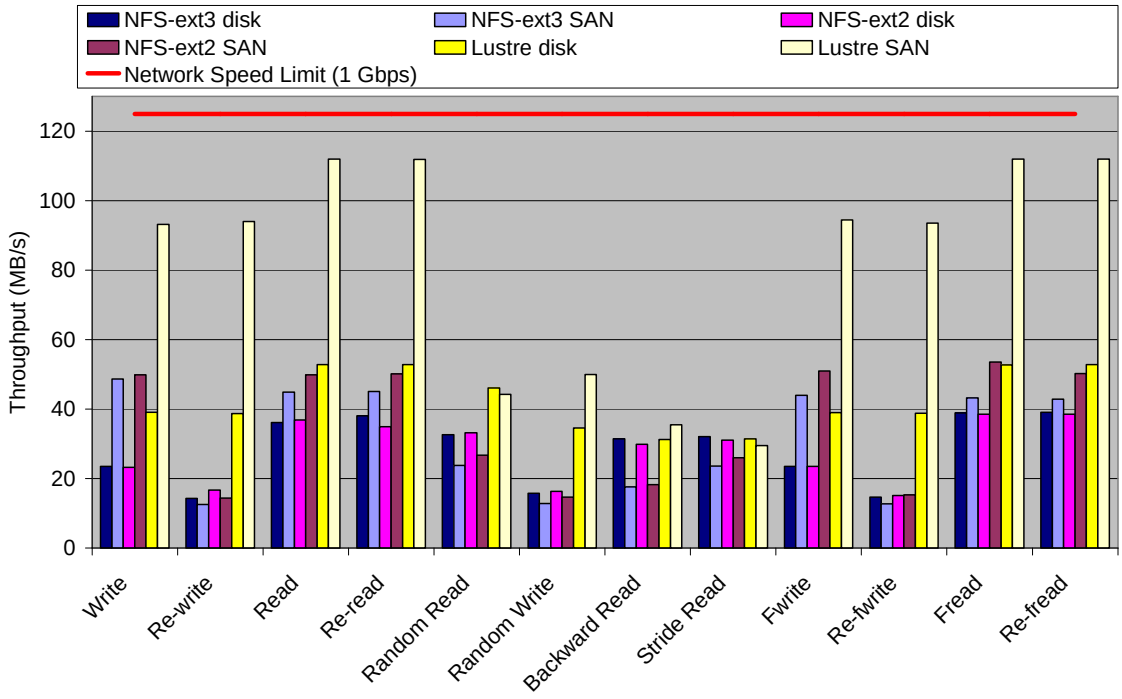

Figure 40 - Remote throughput

For sequential actions, the SAN storage is always at least twice faster than the internal disk... taking in account that for Lustre, the SAN platform performances are limited by the network, so its performances could be even better.

For non-sequetial actions, the disk seems to be a little bit faster for NFS. Writing on Lustre is performed faster by the SAN, and reading on Lustre gives mixed results.

Disk results for Lustre are the same that we observed locally. This is what we expected, because local tests gave results inferior to 60 MB/s, hence the Gigabit network – which is the only constraint we added – has no impact on that test.

1.4. Access time

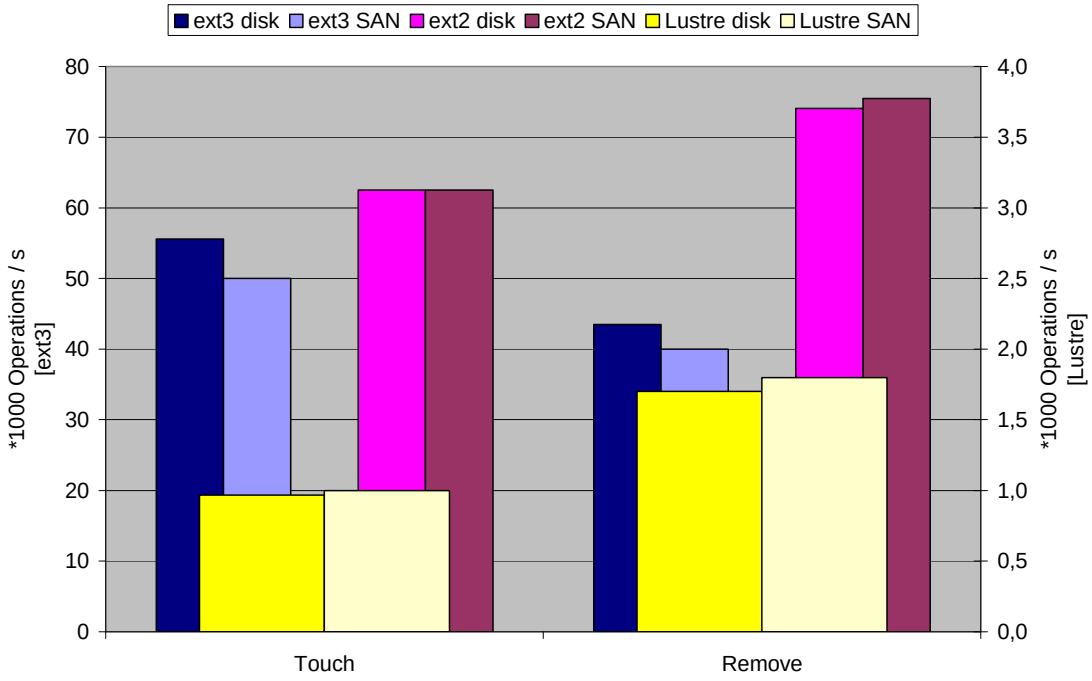

Figure 41 - Local access times

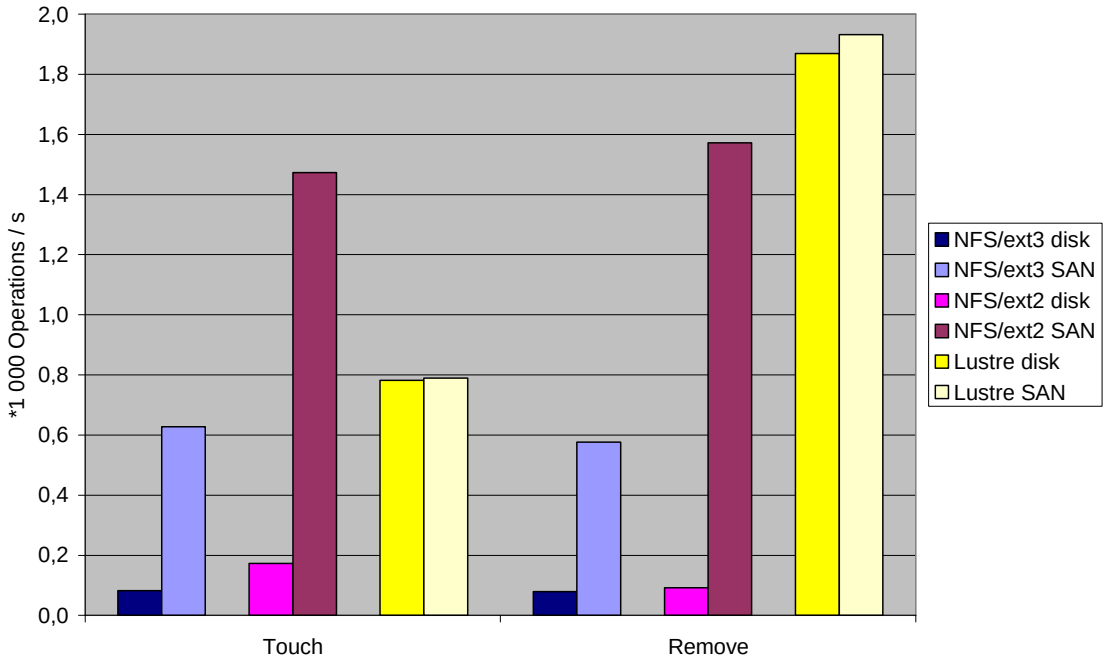

Figure 42 - Remote access times

In opposition to throughput tests, using whether the disk or the SAN has no significant impact on local access times, whatever the file system.

On Lustre, remote results are equivalent to local ones. On the contrary, NFS has a big impact on ext2 and ext3 performances, almost when data are stored on the internal server's disk. Indeed, NFS/ext3 is 6 times faster on the SAN than on the disk, and NFS/ext2 is up to 20 times faster on the SAN.

Unexpectedly, when data are stored on server's local disk, NFS is more than 500 times slower than ext3 ! Hence, the usage of the internal disk should be strongly avoided.

2. Comparing client and server machines

In this test, we are going to compare throughput and access time performances of both server and client machines. So we will perform these tests on the ext2 and ext3 file systems, with the internal disk as storage device. It would have been interesting to perform the same tests with a SAN, but it was not possible, since the SAN was not installed for the client machines.

The results show that server machines are more efficient than client ones, whatever the file system – ext2 or ext3. Indeed, as we can see on figures 7 and 8, the throughput is around 30% higher on the server than on the client, and the access time is always better on the server – up to 60% better on ext2.

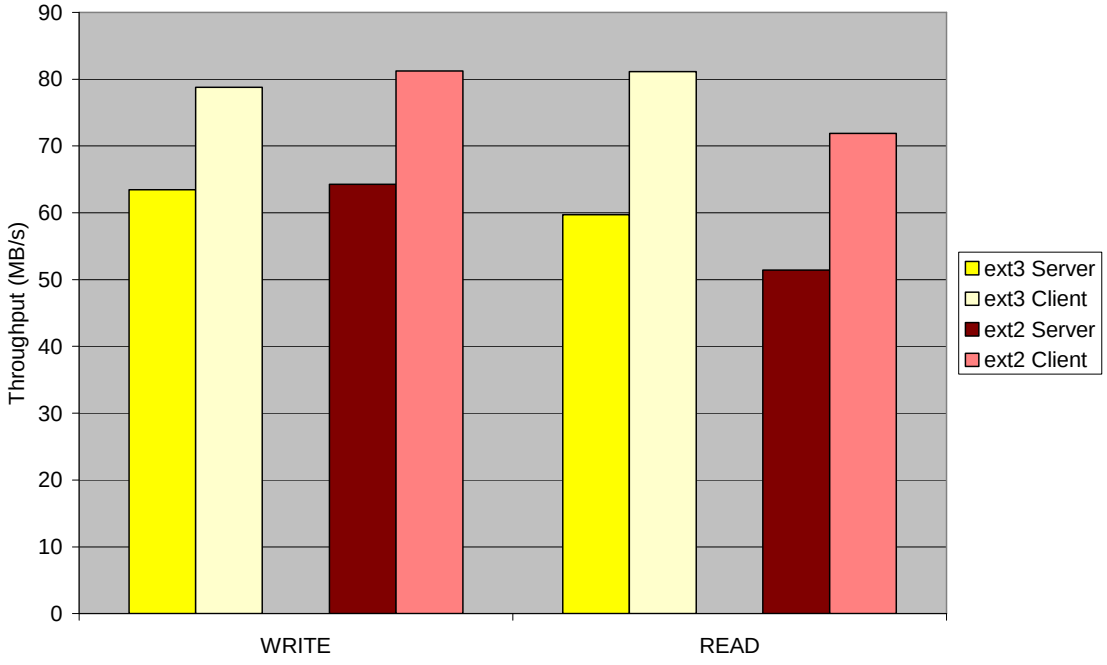

Figure 43 – Comparing disk throughput between client and server

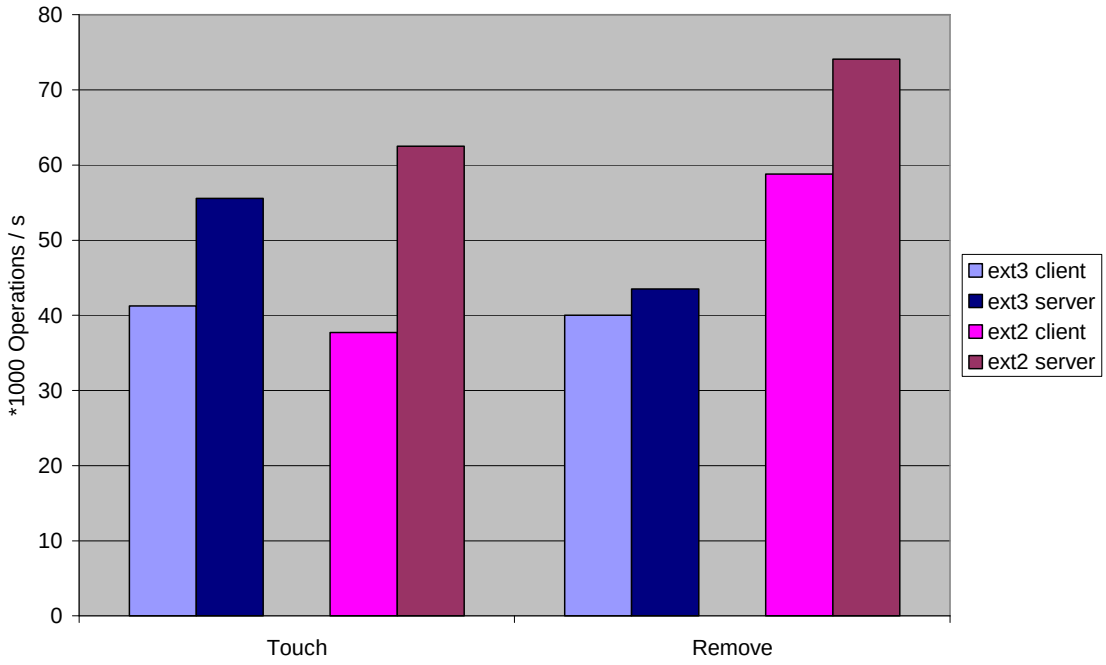

Figure 44 - Comparing disk access times between client and server